



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)**

Колледж экономики, управления и права

**Методические рекомендации
для практических занятий и организации
самостоятельной работы студентов
по МДК.08.01 Проектирование и разработка интерфейсов пользователя**

Специальность
09.02.07 Информационные системы и программирование

Методические рекомендации по ПМ.03 Разработка и интеграция программного обеспечения МДК.08.01 Проектирование и разработка интерфейсов пользователя разработаны с учетом ФГОС среднего профессионального образования специальности 09.02.07 Информационные системы и программирование, предназначены для студентов и преподавателей колледжа.

Методические рекомендации содержат цели изучения курса, задачи курса, методы обучения, планируемые результаты, содержание МДК, необходимый теоретический материал, команды для разметки и оформления сайта, а также список рекомендуемой литературы.

Составитель (автор): С.В.Шинакова, преподаватель колледжа ЭУП

Рассмотрены на заседании предметной (цикловой) комиссии специальностей 09.02.07 Информационные системы (по отраслям), 09.02.05 Прикладная информатика (по отраслям) и 09.02.07 Информационные системы и программирование

Протокол № 9 от «30» июня 2023 г

Председатель П(Ц)К специальности  С.В.Шинакова

и одобрены решением учебно-методического совета колледжа.

Протокол № 7 от «30» июня 2023 г

Председатель учебно-методического совета колледжа


личная подпись С.В.Шинакова

Рекомендованы к практическому применению в образовательном процессе.

СОДЕРЖАНИЕ

Задачи курса.....	5
Планируемые результаты курса.....	5
Тематика практических и самостоятельных работ по разделу 1 Введение.....	6
Практическое занятие №1-2 Разработка сайта средствами Конструкторов	6
Тематика практических и самостоятельных работ по разделу 2 Web-дизайн.....	11
Практическое занятие № 3 Разработка ТЗ, определение цветовой схемы сайта.....	11
Практическое занятие № 4 Взаимодействие пользователя с сайтом.....	11
Тематика практических и самостоятельных работ по разделу 3 Основы web-технологий.....	14
Практическое занятие № 5 Разметка текста в HTML.....	14
Практическое занятие № 6-7 Списки и изображения в HTML.....	30
Часть 1 Списки	30
Часть 2 Изображения в HTML.....	34
Практическое занятие № 8-9 Ссылки в HTML	37
1 Абсолютный и относительный URL.....	38
2 Гиперссылки в пределах одного документа.....	39
3 Ссылка на почтовый ящик	40
Практическое занятие №10 Таблицы	41
Основные атрибуты тега <TABLE>	42
Выравнивание данных в ячейках	43
Объединение ячеек	43
Цвет в таблицах.....	45
Практическое занятие № 11 – 12 Формы в HTML-документах	46
1 Тег <FORM>	47
2 Работа с тегами форм	47
3 Тег <TEXTAREA>	48
4 Тег <SELECT>.....	48
5 Тег <INPUT>	50
6 Тип поля ввода, атрибут TYPE.....	51
7 Нестандартное использование элементов форм	55
Практическое занятие №13 Фреймы	58
1 Контейнер <FRAMESET>	58
2 Определение параметров кадров	58
3 Тег <FRAME>.....	61
4 Организация ссылок.....	63
Тематика практических и самостоятельных работ по разделу 3 CSS.....	66
Практическое занятие №14 Оформление текста с помощью CSS	67
Задание свойств шрифтов	67
Задание свойств текста	70
Практическое занятие №15 Цвет и фоновое изображение CSS	73
Практическое занятие №16 Модель компоновки	79
Практическое занятие №17 Оформление ссылок и списков CSS	84
Оформление списков	84
Оформление ссылок.....	86

Оформление состояния ссылок	86
Практическая работа №18 Создание таблиц на сайте	90
Практическое занятие №19 Позиционирование в CSS	99
Тематика практических и самостоятельных работ по разделу 4 HTML 5	105
Практическое занятие №20-21 Применение HTML5 для разработки индивидуального сайта ..	105
Тематика практических и самостоятельных работ по разделу 5 Java Script	106
Java Script.....	107
Часть 1 Введение в JavaScript. Программное взаимодействие с HTML документами на основе DOM API.	107
Элементы языка JavaScript.....	107
Структура сценария	107
Переменные	107
Объекты	108
Операции.....	108
HTML DOM.....	110
Диалоговые элементы.....	113
Часть 2 Клиентские сценарии. Использование регулярных выражений	115
Обработка событий в JavaScript	115
Регулярные выражения	117
Использование регулярных выражений в JavaScript.....	119
Приложение А_Статья из книги "Физики смеются. Но смеются не только физики", М.: Совпадение, 2020.	123
Приложение Б_Фрагмент из книги "Полное собрание законов Мерфи", Мн.: ООО "Попурри", 2005	125
Приложение В_Статья из книги "Физики смеются. Но смеются не только физики", М.: Совпадение, 206.	127

Задачи курса

Основными задачами курса являются:

- познакомить с видами веб-сайтов, их функциональными, структурными и технологическими особенностями;
- сформировать навыки элементарного проектирования и конструирования веб-сайта;
- сформировать представление о языке HTML, о CSS и научить использовать их для создания и оформления веб-страниц;
- сформировать навыки веб-программирования средствами JavaScript;
- сформировать навыки коллективной работы с комплексными веб-проектами.

Планируемые результаты курса

В рамках курса обучающиеся овладевают следующими знаниями, умениями и компетенциями.

Знать:

- нормы и правила выбора стилистических решений;
 - современные методики разработки графического интерфейса;
 - требования и нормы подготовки и использования изображений в информационно-телекоммуникационной сети
- "Интернет" (далее - сеть Интернет); государственные стандарты и требования к разработке дизайна веб-приложений.

Уметь:

- создавать, использовать и оптимизировать изображения для веб-приложений;
 - выбирать наиболее подходящее для целевого рынка дизайнерское решение;
- создавать дизайн с применением
- промежуточных эскизов, требований к эргономике и технической эстетике;
 - разрабатывать интерфейс пользователя для веб-приложений с использованием современных стандартов.

Качество продукции оценивается следующими способами:

- по количеству творческих элементов в сайте;
- по степени его оригинальности;
- по относительной новизне сайта;
- по емкости и лаконичности созданного сайта, по его интерактивности;
- по практической пользе сайта и удобству его использования.

Тематика практических и самостоятельных работ по разделу 1 Введение

Тема работы	Вид работы	Содержание занятия	Количество часов
Разработка сайта средствами конструкторов	Практическое занятие	Занятие № 1-2 Разработка сайта средствами Конструкторов	4
Разработка сайта средствами конструктора по индивидуальной теме	Самостоятельная работа		4

Практическое занятие №1-2 Разработка сайта средствами Конструкторов

Занятие № 1 «Создание макета сайта гостиницы»

Цели работы:

1. Формирование навыков работы с web-сайтами.
2. Формирование умений создавать макет сайта с помощью возможностей конструкторов сайтов.

Задание: Подготовить и сохранить в отдельной папке «Сайт» материал для работы:

1. Логотип гостиницы
2. Фотографии гостиницы:
 - 2-3 Фото здания (в графическом редакторе на изображении поместить полупрозрачный логотип гостиницы)
 - 5-6 Фото любого помещения гостиницы (в графическом редакторе на изображении поместить полупрозрачный логотип гостиницы)
3. Фотографии сотрудников (в графическом редакторе на изображениях поместить полупрозрачный логотип гостиницы):
 - Директор
 - Зам. директора по работе с клиентами
 - Зам. директора по развитию персонала
 - Администратор
 - Горничная
 - Горничная (либо любой другой обслуживающий персонал)
4. Карту с расположением гостиницы (Скопировать из Интернета) (в графическом редакторе на изображении поместить полупрозрачный логотип гостиницы)
5. Фотографии услуг гостиницы (room-service) (в графическом редакторе на изображениях поместить полупрозрачный логотип гостиницы)

Технология работы:

1. Открыть **он-лайн конструктор** <https://www.wix.com>
2. Выполнить регистрацию на сайте
3. Выбрать **категорию Отели > Тип отеля** (соответственно варианту)

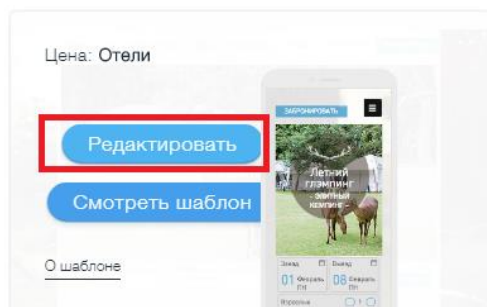
Категории

Все шаблоны

- > Бизнес
- > Интернет-магазин
- > Фото
- > Видео
- > Музыка
- > Дизайн
- > Рестораны и еда
- > **Отели**
 - Отели
 - Аренда жилья
 - В&В
 - Хостелы и кемпинги
 - Путешествия и туризм

4. Выбрать **подходящий шаблон** и нажать кнопку **РЕДАКТИРОВАТЬ**

Выберите шаблон в категории Хостелы и кемпинги

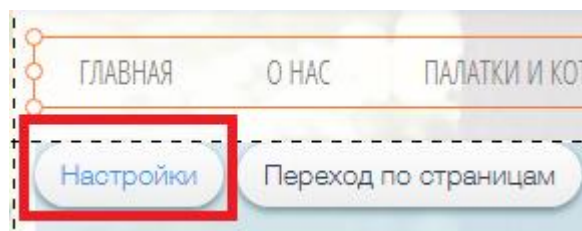


Кемпинг

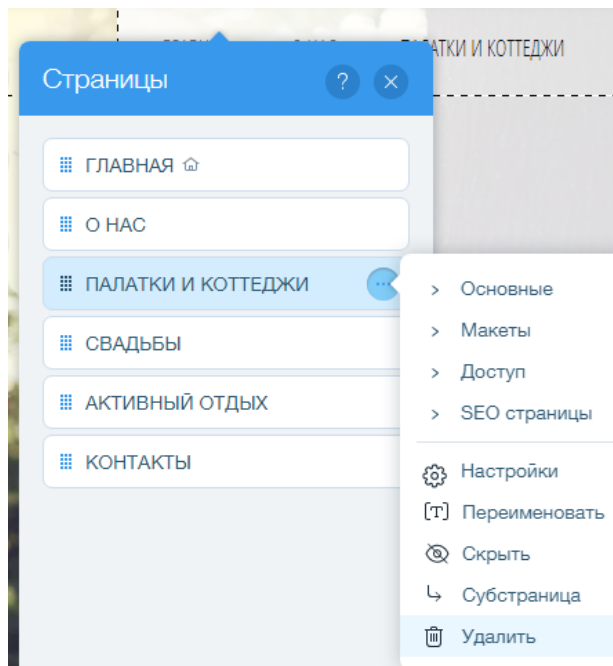


Горнолыжный курорт

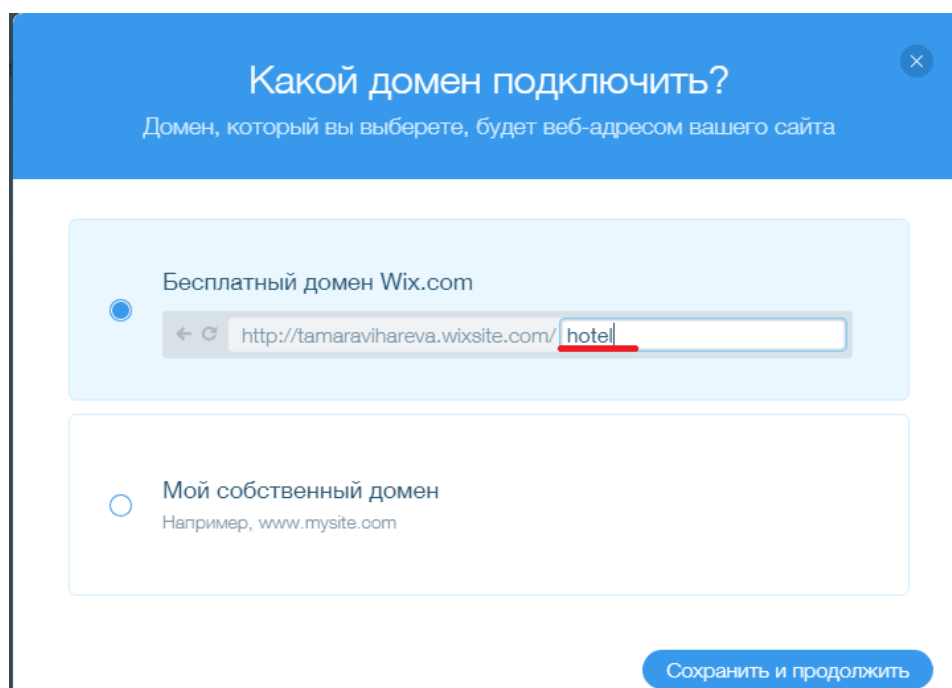
6. В появившемся шаблоне отредактировать верхнее меню будущего сайта:
1. Щелкнуть по области с названиями будущих страниц сайта
 2. Нажать кнопку Настройки:



7. Отредактировать структуру будущего сайта – в итоге должно получиться 5 страниц **«Главная», «О нас», «Наши координаты», «Номера», «Сотрудники»:**
1. Кнопкой «Удалить» избавьтесь от лишних страниц;
 2. Кнопкой «Переименовать» назовите страницы соответственно заданию

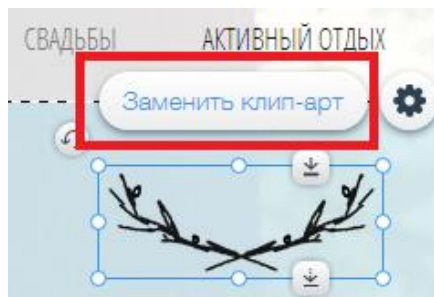


8. Сохранить макет, используя имя вашей гостиницы (**Сайт > Сохранить**):

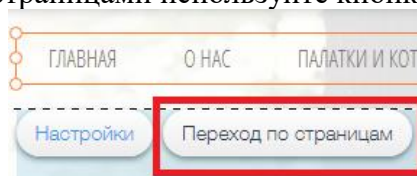


9. Заполните страницы **«Главная»**, **«О нас»**, **«Наши координаты»**, **«Сотрудники»** своими материалами.

1. Установить логотип гостиницы на всех страницах сайта, нажав кнопку «Заменить клип-арт»:



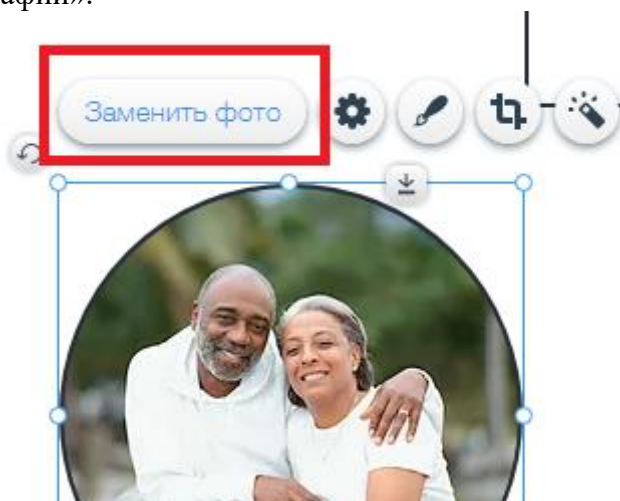
2. Для перехода между страницами используйте кнопку «Переход по страницам»:



3. Для редактирования текста на страницах используйте кнопку «Редактировать текст»:



4. Для добавления и редактирования изображений используйте кнопку «Заменить фото» или «Заменить фотографии»:



10. Заполните страницы **«Номера»** своими материалами.
 11. Для предварительного просмотра макета в браузере используйте команду из области задач **Просмотр веб-узла**
 12. В отдельно созданной папке «Макет» выполнить сохранение макета в качестве фильтрованной HTML-версии (это предпоследний шаг перед загрузкой в Интернет);
 Для этого выполнить **Файл → Поместить на веб-узел...** открыть созданную папку «Макет», нажать **Сохранить** (Имя файла **Index** предлагается не изменять)

Итог работы:

Представить преподавателю HTML-версию созданного макета сайта гостиницы.

Занятие № 2

Создание сайта по индивидуальной теме.

1. Рекламное агентство
2. Туристическое агентство
3. Агентство недвижимости
4. Школа по изучению иностранных языков
5. Магазин компьютерной техники
6. Магазин сантехники
7. Автосалон
8. Шиномонтаж
9. Фирма по доставке питьевой воды
10. Ремонт компьютерной техники
11. Интернет-магазин обуви
12. Продажа картин
13. Свадебный салон
14. Фотоателье
15. Строительная компания
16. Заказ билетов
17. Грузоперевозки
18. Система охраны
19. Салон красоты
20. Пиццерия
21. Автозвук
22. Магазин одежды
23. Страйкбольный магазин

Итог работы:

Представить преподавателю созданный сайт.

Самостоятельная работа

Разработка сайта средствами конструктора по индивидуальной теме.

Тематика практических и самостоятельных работ по разделу 2 Web-дизайн

Тема работы	Вид работы	Содержание занятия	Количество часов
Разработка ТЗ, определение цветовой схемы сайта	Практическое занятие	Занятие № 3 Разработка эскизов веб-приложения. Разработка прототипа дизайна веб-приложения	2
Взаимодействие пользователя с сайтом	Практическое занятие	Занятие № 5 Разработка схемы интерфейса веб-приложения. Провести тестирование, проверку 3 сайтов на соответствие веб-стандартам.	2

Практическое занятие № 3 Разработка ТЗ, определение цветовой схемы сайта

Цель занятия: формировать знания цветовых схем; навык по разработке технической документации по разработке сайтов.

Задание: Разработать техническое задание на разработку сайта по индивидуальной теме. Сайт должен содержать не менее 4 страниц.

Итог работы:

Представить преподавателю ТЗ для разработки многостраничного сайта по индивидуальной теме.

Практическое занятие № 4 Взаимодействие пользователя с сайтом

Цель занятия: формировать навык по разработке прототипа дизайна веб-приложения.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

UX дизайн – это проектирование интерфейса на основе исследования пользовательского опыта и поведения.

При разработке UX дизайна приложений, мы следуем следующим принципам:

- Простота
- Интуитивность
- Гибкость
- Наглядность
- Предотвращение и устранение ошибок

Процесс работы над UX дизайном проходит в несколько этапов:

1 Исследование

Наши аналитики проводят сбор информации о продукте и клиенте, учитывают статистику использования текущего интерфейса, анализируют устройства потенциальных пользователей.

2 Проектирование структуры

Создание разметки рабочей области, активных и неактивных зон; определение количества экранов/страниц и вкладок, их содержания

3 Написание пользовательских сценариев

На основе предоставленного описания работы интерфейса мы создаем список задач, которые пользователь сможет выполнять через интерфейс

4 Постановка задач для разработчиков и дизайнеров

Полученные схемы с детализацией функциональных возможностей будущего интерфейса и учетом пользовательских действий передаются UI дизайнерам для создания целостного визуального решения.

При **разработке UI дизайна** нашей целью является создание максимально простого и продуктивного взаимодействия с точки зрения достижения целей пользователя. Для этого мы тщательно прорабатываем единый стиль и логику взаимодействия элементов интерфейса.

Основным критерием создания качественного UI дизайна является *удобство использования*. При разработке UI дизайна необходимо придерживаться рекомендаций и признанных стандартов качества и стремиться к созданию ясности, лаконичности, логичности и интуитивной понятности интерфейса любому пользователю (как новому, так и знакомому с приложением).

Разработка пользовательского интерфейса включает в себя проработку и реализацию следующих задач:

- Создание UI элементов (иконки, кнопки, формы)
- Разработка стилистического решения
- Определение цветовой палитры
- Расположение элементов интерфейса на экране

ХОД РАБОТЫ

Задание 1. «Основы UX-дизайна: создание прототипа сайта».

- 1 Нарисуйте простой и схематичный прототип вашего сайта карандашом.
 - Отметьте основные блоки, детализировать не обязательно
 - Можно написать заголовки
 - Прорисуйте интерактивные элементы — кнопки, меню и т. д.
- 2 Убедитесь, что получившийся прототип
 - решает задачу
 - может достичь цели
 - соответствует персонам и целям
- 3 Проверьте прототип коридорным тестированием
 - Покажите прототип нескольким людям. Не важно, из вашей ли целевой аудитории (ЦА) или нет: сейчас мы ищем самые заметные ошибки.
 - Попросите их вслух прокомментировать, что они видят.
 - Дайте им задание. Например, «Как со мной связаться?». И попросите рассказать, как они будут его решать
- 4 Исправьте ошибки и детализируйте прототип.
 - Можно это делать на компьютере в удобном вам инструменте: Word, Figma, Miro
 - Добавьте примерные тексты

- Можно добавить графики и картинки, несущие информацию
- Протестируйте на новых людях, желательно из целевой аудитории

Задание 2. Разработка схемы интерфейса веб-приложения.

User flow — карта навигации, по которой видно поведение пользователя приложения, как он достигает цели и как легко ему это удаётся. Внешне User flow выглядит как логически связанные друг с другом прямоугольники, акцент в которых сделан на действиях пользователя. Для разработки карты навигации можно использовать веб-сервис Overflow.

Задание 3. Провести тестирование, проверку 3 сайтов на соответствие веб-стандартам.

Итог работы:

Представить преподавателю для разработанные материалы.

Тематика практических и самостоятельных работ по разделу 3 Основы web-технологий

Тема работы	Вид работы	Содержание занятия	Количество часов
Разметка текста в HTML	Практическое занятие	Занятие № 5 Логическое и Физическое форматирование текста	2
Списки и изображения в HTML	Практическое занятие	Занятие № 6-7 Оформления информации в виде списков. Работа с графическими изображениями. Форматы графических файлов. Включение изображений в HTML.	4
Ссылки в HTML	Практическое занятие	Занятие № 8-9 Создание гиперссылок. Организация навигации на сайте. Создание навигационного меню. Различные типы меню. Карты-изображения	4
Таблицы	Практическое занятие	Занятие № 10 Работа с таблицами и ссылками в HTML. Рубежный контроль	2
Формы	Практическое занятие	Занятие № 11-12 Разработка сайта с использованием форм. Назначение формы в HTML-документе. Контейнер размещения формы. Поле ввода текста. Поле ввода пароля. Флажки. Разработка сайта с использованием форм. Радиокнопки. Командные кнопки. Многострочная текстовая область. Раскрывающиеся списки. Обработка заполненных форм.	4
Фреймы	Практическое занятие	Практическое занятие №13 Разработка сайта с использованием фреймов	2

Практическое занятие № 5 Разметка текста в HTML

Цель занятия: формировать навык HTML-разметки применяя логическое и физическое форматирование.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Часть 1

1. Основы языка HTML

HyperText Markup Language (HTML) -- язык разметки гипертекста -- предназначен для написания гипертекстовых документов, публикуемых в World Wide Web.

Гипертекстовый документ -- это текстовый файл, имеющий специальные метки, называемые тегами, которые впоследствии опознаются браузером и используются им для отображения содержимого файла на экране компьютера. С помощью этих меток можно выделять заголовки документа, изменять цвет, размер и начертание букв, вставлять графические изображения и таблицы. Но основным преимуществом гипертекста перед обычным текстом является возможность добавления к содержимому документа **гиперссылок** -- специальных конструкций языка HTML, которые позволяют щелчком мыши перейти к просмотру другого документа.

Создание и разметка документа при помощи обычного редактора текста (таких, как блокнот, NotePad++ и др.). При этом способе в текст вручную вставляются команды языка HTML.

Как уже отмечалось, HTML-документ содержит символьную информацию. Одна ее часть -- собственно текст, т. е. данные, составляющие содержимое документа. Другая -- **теги**, -- специальные конструкции языка HTML, используемые для разметки документа и управляющие его отображением. Именно теги языка HTML определяют, в каком виде будет представлен текст, какие его компоненты будут исполнять роль гипертекстовых ссылок, какие графические или мультимедийные объекты должны быть включены в документ. Графическая и звуковая информация, включаемая в HTML-документ, хранится в отдельных файлах. Программы просмотра HTML-документов (браузеры) интерпретируют флаги разметки и располагают текст и графику на экране соответствующим образом. Для файлов, содержащие HTML-документы, приняты расширения .htm или .html.

Прописные и строчные буквы при записи тегов не различаются. В большинстве случаев теги используются парами. Пара состоит из открывающего и закрывающего тегов. Синтаксис открывающего тега:

`<имя_тега [атрибуты]>`

Прямые скобки, используемые в описании синтаксиса, означают, что данный элемент может отсутствовать. Имя закрывающего тега отличается от имени открывающего лишь тем, что перед ним ставится наклонная черта:

`</имя_тега>`

Атрибуты тега записываются в следующем формате:

`имя[="значение"]`

Кавычки при задании значения аргумента не обязательны и могут быть опущены. Для некоторых атрибутов значение может не указываться. У закрывающего тега атрибутов не бывает.

Действие любого парного тега начинается с того места, где встретился открывающий тег и заканчивается при встрече соответствующего закрывающего тега. Часто пару, состоящую из открывающего и закрывающего тегов, называют **контейнером**, а часть текста, окаймленную открывающим и закрывающим тегом, -- **элементом**.

Последовательность символов, составляющая текст, может состоять из пробелов, табуляций, символов перехода на новую строку, символов возврата каретки, букв, знаков

препинания, цифр, и специальных символов (например, +, #, \$, @), *за исключением* следующих четырех символов, имеющих в HTML специальный смысл: < (меньше), > (больше), & (амперсанд) и " (двойная кавычка). Если необходимо включить в текст какой-либо из этих символов, то следует закодировать его особой последовательностью символов.

К специальным символам можно отнести и неразрывный пробел. Использование этого символа - один из способов увеличить расстояние между некоторыми словами в тексте. Обычные пробелы использовать для этих целей нельзя, так как группа подряд идущих пробелов интерпретируется браузером как один.

Каждая из таких зарезервированных последовательностей начинается символом амперсанда (&) и заканчивается точкой с запятой (;).

Последовательность	Символ
<	символ <
>	символ >
&	символ &
"	символ " (кавычка)
 	неразрывный пробел

2. Структура HTML-документа

Самым главным из тегов HTML является одноименный тег - <HTML>. Он должен всегда открывать ваш документ, так же, как тег </HTML> должен непременно стоять в последней его строке. Эти теги обозначают, что находящиеся между ними строки представляют единый гипертекстовый документ. Без этих тегов браузер или другая программа просмотра не в состоянии идентифицировать формат документа и правильно его интерпретировать.

Закрывающий тег так же важен, как и открывающий. Если, например, документ включен в электронное письмо, тег </HTML> дает команду программе просмотра прекратить интерпретацию текста, как HTML-кода.

HTML-документ состоит из двух частей: **заголовок** (head) и **тело** (body), расположенных в следующем порядке:

```
<HTML>
<HEAD>
...
</HEAD>

<BODY>
...
</BODY>
</HTML>
```

В HTML-документ можно включать *комментарии*, позволяющие скрыть часть текста от браузера. Все, что заключено между последовательностями символов <!-- и -->, при просмотре страницы остается невидимым. Комментарии не могут быть вложенными друг в друга.

3. Заголовок документа

Включение в документ заголовочной части не является обязательным. Задачей заголовка является представление необходимой информации для браузера и сервера HTTP. Информация, размещенная внутри заголовка документа, обычно не выводится на экран (кроме названия документа).

Заголовок документа открывается тегом `<HEAD>`, который обычно следует сразу же за тегом `<HTML>`. Закрывающий тег `</HEAD>` показывает конец этого раздела, между ними располагаются остальные теги заголовка документа.

Чаще всего в заголовок документа включают парный тег `<TITLE> ... </TITLE>`, определяющий **название** документа. Многие программы просмотра используют его как заголовок окна, в котором выводят документ. Программы, индексирующие документы в сети Интернет, используют название для идентификации страницы. Хорошее название должно быть достаточно длинным для того, чтобы можно было корректно указать соответствующую страницу, и в то же время оно должно помещаться в заголовке окна. Название документа вписывается между открывающим и закрывающим тегами.

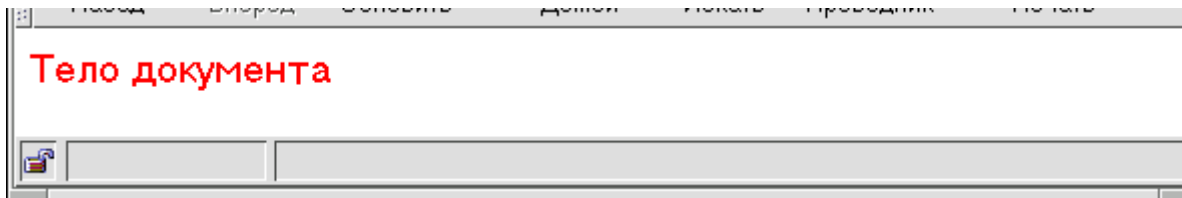
4. Тело документа

В отличие от заголовка, тело документа является обязательным элементом, так как в нем располагается весь материал вашего документа (есть только одно исключение, с которым мы познакомимся далее, - когда документ содержит вместо тела группу фреймов). Тело документа размещается между тегами `<BODY>` и `</BODY>`. Все, что размещено между этими тегами, интерпретируется браузером в соответствии с правилами языка HTML, позволяющими корректно отображать страницу на экране монитора.

Тег `<BODY>` не только обозначает начало содержимого документа, но и задает его основные свойства: цвет фона, текста и многое другое. Эти свойства определяются с помощью атрибутов, которые приведены в таблице.

Атрибут	Назначение
<code>ALINK</code>	Определяет цвет активной ссылки
<code>BACKGROUND</code>	Указывает на URL-адрес изображения, которое используется в качестве фонового
<code>BGCOLOR</code>	Определяет цвет фона документа
<code>LINK</code>	Определяет цвет непосещенной ссылки
<code>TEXT</code>	Определяет цвет текста
<code>VLINK</code>	Определяет цвет посещенной ссылки

Пример 4.1



```

<HTML>
<HEAD>
<TITLE>Моя страничка</TITLE>
</HEAD>
<BODY TEXT=red BGCOLOR=white>
Тело <!-- Это комментарий --> документа
</BODY>
</HTML>

```

5. Цветовое оформление документа

Определение цвета составных частей документа - один из первых шагов в его создании. Если этого не сделать, то будут использоваться цвета по умолчанию, определяемые установками браузера. Не существует каких-либо правил создания хорошо сбалансированной цветовой палитры документа. Нужно лишь заботиться о том, чтобы можно было прочитать текст, не испытывая при этом неудобств. При выборе цветовой палитры старайтесь поддерживать высокую контрастность текста и фона и избегайте соседства областей с близкими цветами.

Цвет может быть задан названием (например, green) или шестнадцатеричным числом, определяющим цвет в модели RGB. Эта цветовая модель базируется на определении цвета как композиции трех основных оттенков цвета: красного (Red), зеленого (Green) и синего (Blue). Каждая компонента задается двузначным шестнадцатеричным числом (т. е. изменяется от 00 до FF). Затем эти значения объединяются в одно число, перед которым ставится символ # (большинство современных браузеров может распознать цвет и без указания символа #).

Следует также отметить, что в записи шестнадцатеричного значения цвета можно использовать как большие, так и маленькие латинские буквы, например, запись #00FF00 равнозначна записи #00ff00.

Ниже представлена таблица 16 стандартных цветов вместе с их шестнадцатеричными кодами. Современные браузеры понимают 140 наименований цветов, часть из которых приведена в главе "Динамический HTML".

Цвет	Код	Цвет	Код
black (черный)	#000000	silver (серебряный)	#C0C0C0
maroon (темно-бордовый)	#800000	red (красный)	#FF0000
green (зеленый)	#008000	lime (известь)	#00FF00
olive (оливковый)	#808000	yellow (желтый)	#FFFF00

navy (темно-синий)	#000080	blue (синий)	#0000FF
purple (фиолетовый)	#800080	fuchsia (фуксия)	#FF00FF
teal (сине-зеленый)	#008080	aqua (аква)	#00FFFF
gray (серый)	#808080	white (белый)	#FFFFFF

Пример 5.1

Чтобы установить синий цвет фона документа нужно для синей составляющей цвета указать максимальное значение, а остальные сделать равными нулю: `<BODY BGCOLOR="#0000FF">`.

Тот факт, что разработчик Web-страницы ничего не знает о компьютере, на котором этот документ будет просматриваться, накладывает дополнительные ограничения на использование RGB-модели. На некоторых мониторах невозможно отобразить все разнообразие оттенков. Браузеры в этом случае сокращают используемое количество цветов, переназначая их под собственные палитры. Использование разработчиками гипердокументов **Web-палитры** является в некотором роде гарантией того, что документ будет выглядеть одинаково на различных дисплеях.

Web-палитра распознает по шесть оттенков красного, синего и зеленого цвета, что в результате дает 216 возможных значений цветов ($6 \times 6 \times 6 = 216$). Поэтому ее иногда называют "куб $6 \times 6 \times 6$ ". В таблице приведены численные значения цветов из Web-палитры.

Шестнадцатеричные	Десятичные	Процентные
00 (самый темный)	0	0 %
33	51	20 %
66	102	40 %
99	153	60 %
CC	204	80 %
FF(самый светлый)	255	100 %

Цвета, RGB-компоненты которых входят в Web-палитру, называют **Web-надежными** цветами. Среди всех цветов, имеющих имена, только 10 цветов входят в Web-палитру: aqua, black, blue, cyan, fuchsia, lime, magenta, red, white и yellow.

6. Разделение текста на абзацы

Язык HTML предполагает, что автор документа ничего не знает о компьютере своего читателя. Читатель вправе установить любой размер окна и пользоваться любым из имеющихся у него шрифтов. Это означает, что место переноса в строке определяется только программой просмотра и установками конечного пользователя. Символы перевода строки оригинального документа игнорируются, в результате чего текст, отлично смотревшийся в окне вашего редактора, может превратиться в сплошной неудобочитаемый текст в окне программы просмотра.

Избежать этой неприятности позволяет разделение на абзацы при помощи тега `<P>`. Разместите его в начало каждого абзаца, и программа просмотра отделит абзацы друг от друга пустой строкой. Использование закрывающего тега `</P>` необязательно. Несколько стоящих подряд тегов `<P>` не дают дополнительного пространства между абзацами.

Тег абзаца имеет один атрибут, поддерживаемый большинством браузеров. Это атрибут `ALIGN`, задающий выравнивание текста в абзаце. Если этот атрибут не задан, то текст выравнивается по левому краю окна браузера. В таблице представлены возможные значения этого атрибута:

Значение	Функция
<code>LEFT</code>	Выравнивание текста по левой границе окна браузера
<code>CENTER</code>	Выравнивание по центру окна браузера
<code>RIGHT</code>	Выравнивание по правой границе окна браузера
<code>JUSTIFY</code>	Выравнивание текста по ширине окна браузера

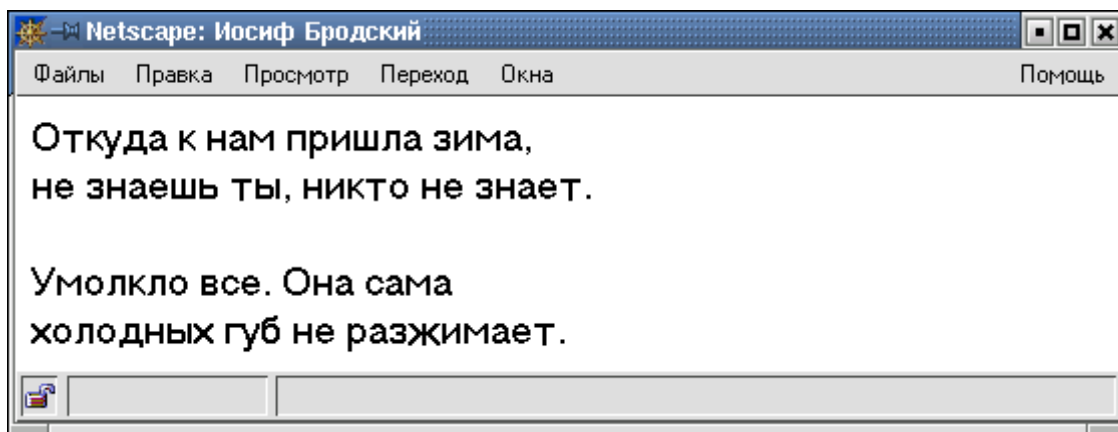
Пример 6.1

Если вы хотите, чтобы текст абзаца был выровнен по центру, нужно написать следующее: `<P ALIGN=CENTER> Текст </P>`

7. Разрыв строки

Иногда требуется "разорвать" текст, перенести его остаток на новую строку, при этом не выделяя нового абзаца. Для этого используется тег разрыва строки `
`. Он заставляет программу просмотра выводить стоящие после него символы с новой строки. В отличие от тега абзаца, тег `
` не добавляет пустую строку. У этого тега нет парного закрывающего тега.

Пример



```

<HTML>
<HEAD>
<TITLE> Иосиф Бродский </TITLE>
</HEAD>
  <BODY TEXT=black BGCOLOR=white>
    Откуда к нам пришла зима, <BR> не знаешь ты, никто не знает.
    <P> Умолкло все. Она сама <BR> холодных губ не разжимает.
  </BODY>
</HTML>

```

Некоторые браузеры интерпретируют несколько стоящих рядом тегов `
` как один тег, поэтому не стоит использовать его для вставки пустых строк.

Бывают случаи, когда возникает надобность в операции противоположного назначения - запретить перевод строки. Текст, заключенный между тегами `<NOBR>` и `</NOBR>`, будет гарантированно располагаться в одной строке без переноса на другую. Горизонтальные линии

Другим методом разделения документа на части является проведение горизонтальных линий. Они визуально подчеркивают законченность той или иной области страницы. Тег `<HR>` позволяет провести рельефную горизонтальную линию в окне большинства программ просмотра. Этот тег не требует закрывающего тега. До и после линии автоматически вставляется пустая строка. Атрибуты тега `<HR>` представлены в таблице.

Атрибут	Назначение
<code>ALIGN</code>	Выравнивает по краю или центру; имеет значения <code>LEFT</code> , <code>CENTER</code> , <code>RIGHT</code>
<code>WIDTH</code>	Устанавливает длину линии в пикселах или процентах от ширины окна браузера; в последнем случае добавляется символ <code>%</code>
<code>SIZE</code>	Устанавливает ширину линии в пикселах
<code>NOSHADOW</code>	Отменяет рельефность линии
<code>COLOR</code>	Указывает цвет линии; используется наименование цвета или шестнадцатеричный код

Задания 1

1. Создайте документ с именем `index.html`, в заголовке которого поместите свои имя и фамилию и установите желтый цвет фона.
2. Включите в документ четыре абзаца текста, демонстрирующие различные случаи выравнивания. Разделите их горизонтальными линейками с различными значениями атрибутов.
3. Примените все теги, описанные выше для своей страницы.

Домашнее задание

1. Создать главную html-страницу для сайта (по выбранной на первом занятии темы).
2. Применить теги, рассмотренные на данном занятии.

Часть 2

8. Форматирование гипертекста

Язык HTML поддерживает как логический (logical), так и физический (physical) стили форматирования содержимого документа. Использование логического форматирования указывает на назначение данного фрагмента текста, а при физическом форматировании досконально задается его внешний вид. По возможности стоит использовать логические стили, так как они позволяют браузеру выбрать наиболее подходящий документу вид. Использование логических стилей также поможет читателю разобраться в структуре документа. Физический стиль используется в основном программами, конвертирующими текстовые файлы, содержащие физическое форматирование, в HTML, так как логическое форматирование документа невозможно выполнить автоматически.

8.1 Логическое форматирование

Хотя язык HTML включает теги для достижения различных шрифтовых эффектов (полужирный шрифт, курсив, подчеркнутый шрифт), не все браузеры их поддерживают. Однако все браузеры поддерживают тот или иной способ выделения текста. Поэтому использование логического форматирования текста в любом случае приведет к выделению программой просмотра различных частей текста и выявит структуру документа.

Говоря о логической разметке текста, можно выделить две основные части:

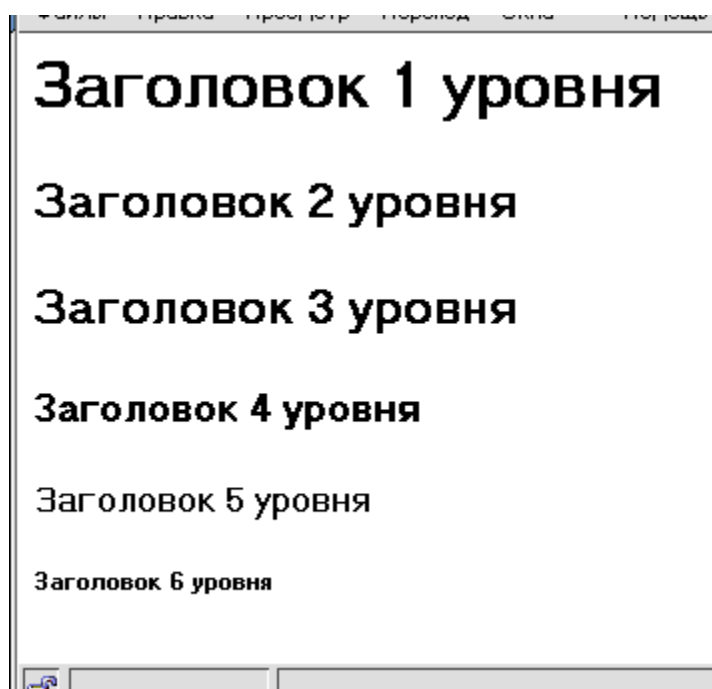
- выделение заголовков в документе;
- логическое выделение элементов текста.

Название документа, задаваемое с помощью тега `<TITLE>`, не выводится на экран как часть документа. Чтобы отобразить название используется один из тегов заголовка. Заголовки в типичном документе разделяются по уровням. Язык HTML позволяет задать шесть уровней заголовков: H1 (заголовок первого уровня), H2, H3, H4, H5 и H6. Заголовок первого уровня имеет обычно больший размер и насыщенность по сравнению с заголовком второго уровня.

Если вы посмотрите на эту главу, то "Логическое форматирование" -- заголовок третьего уровня, "Форматирование гипертекста" -- второго, а "Основы языка HTML" -- первого. На практике заголовки четвертого и далее уровней встречаются лишь в очень больших документах.

Пример 8.1

```
<HTML>
<HEAD>
  <TITLE> Заголовки
</TITLE>
</HEAD>
<BODY BGCOLOR=white>
  <H1> Заголовок 1
уровня</H1>
  <H2> Заголовок 2
уровня</H2>
  <H3> Заголовок 3
уровня</H3>
  <H4> Заголовок 4
уровня</H4>
  <H5> Заголовок 5
уровня</H5>
  <H6> Заголовок 6
уровня</H6>
</BODY>
</HTML>
```



Помните, что если вы забудете поставить закрывающий тег заголовка, вид страницы будет искажен: любой из тегов заголовка автоматически вставляет пустую строку до и после себя.

Теги заголовков поддерживают атрибут **ALIGN**, действие которого аналогично действию такого же атрибута тега выделения абзаца.

Элементы логического выделения фрагментов текста, а также возможное оформление каждого из них приведены в таблице.

Теги	Применение	Результат
<code><CITE></code> <code></CITE></code>	<code><CITE></code> Используется для выделения цитат или названий книг и статей <code></CITE></code>	<i>Используется для выделения цитат или названий книг и статей</i>
<code><CODE></code> <code></CODE></code>	<code><CODE></code> Применяется для вывода небольшого куса программного кода <code></CODE></code>	Применяется для вывода небольшого куса программного кода
<code></code> <code></code>	<code></code> Используется для выделения важных фрагментов текста <code></code>	<i>Используется для выделения важных фрагментов текста</i>
<code><KBD></code> <code></KBD></code>	<code><KBD></code> Выделяет текст, вводимый пользователем с клавиатуры <code></KBD></code>	Выделяет текст, вводимый пользователем с клавиатуры
<code><SAMP></code> <code></SAMP></code>	<code><SAMP></code> Используется для выделения текста примера <code></SAMP></code>	Используется для выделения текста примера
<code></code>	<code></code> Используется для	Используется для выделения

<code></code>	выделения очень важных фрагментов текста <code></code>	очень важных фрагментов текста
<code><VAR></code> <code></VAR></code>	<code><VAR></code> Используется для отметки имен переменных <code></VAR></code>	<i>Используется для отметки имен переменных</i>
<code><STRIKE></code> <code></STRIKE></code>	<code><STRIKE></code> Используется для отметки удаленного текста <code></STRIKE></code>	Используется для отметки удаленного текста

Вы, наверное, обратили внимание на то, что некоторые элементы логической разметки текста дают одинаковый результат. Зачем же тогда нужно такое их разнообразие? Ответ на этот вопрос содержится в названии этой группы тегов. Они предназначены для расстановки логических ударений, выделения логических частей и подчеркивания сути высказываний. Их использование весьма актуально, поскольку, вероятно, в ближайшем будущем станет возможен, например, поиск цитат в Web-пространстве, а, может быть, следующее поколение браузеров научится читать документы вслух. Программы, умеющие распознавать логические ударения, заменят монотонные речевые процессоры сегодняшнего дня.

Для выделения длинных цитат из основного текста в HTML существует тег `<BLOCKQUOTE>`. Этот элемент является контейнером и может содержать любые форматирующие теги.

Современные браузеры реагируют на элемент `<BLOCKQUOTE>` смещением текста цитаты вправо. Некоторые текстовые программы просмотра обозначают цитату символами `>`, располагающимися в крайнем левом столбце экрана. Так как сегодня большинство браузеров являются графическими программами, элемент `<BLOCKQUOTE>` позволяет авторам внести в текст некоторое визуальное разнообразие.

8.2 Физическое форматирование

Одним из отличий HTML-документа от документа, подготовленного на печатной машинке, является возможность форматирования текста. Язык HTML позволяет автору документа выбрать понравившийся ему шрифт, подходящий размер букв, их цвет и начертание. За все эти параметры отображения текста отвечают теги физического форматирования. Они действуют на все символы, стоящие между открывающим и закрывающим тегами.

Теги	Применение	Результат
<code> </code>	<code>Полужирный</code>	Полужирный
<code><I> </I></code>	<code><I>Курсив</I></code>	<i>Курсив</i>
<code><U> </U></code>	<code><U>Подчеркнутый</U></code>	<u>Подчеркнутый</u>
<code><TT> </TT></code>	<code><TT>Пишущая машинка</TT></code>	Пишущая машинка
<code><S> </S></code>	<code><S>Зачеркнутый</S></code>	Зачеркнутый
<code><BIG> </BIG></code>	<code><BIG>Большой</BIG></code>	Большой

<code><SMALL> </SMALL></code>	<code><SMALL>Маленький</SMALL></code>	Маленький
<code><SUP> </SUP></code>	Верхний -- <code>x<SUP>индекс</SUP></code>	Верхний -- $x^{\text{индекс}}$
<code><SUB> </SUB></code>	Нижний -- <code>x<SUB>индекс</SUB></code>	Нижний -- $x_{\text{индекс}}$

Элементы физического форматирования могут быть вложенными друг в друга, хотя конечный результат зависит от браузера. При этом нужно внимательно следить, чтобы один контейнер находился целиком в другом контейнере, например,

`<U>жирный и подчеркнутый текст</U>`

Кроме вышеперечисленных тегов в документе может использоваться тег ``, позволяющий непосредственно задать размер и цвет шрифта. Элемент `FONT` представляет собой контейнер, т. е. требует как открывающего, так и закрывающего тегов, и сам может использоваться внутри любого другого текстового контейнера.

После стартового тега обязательно указание атрибутов, без которых элемент не оказывает никакого влияния на текст, помещенный в контейнер.

Атрибут `FACE` позволяет указать тип шрифта, которым программа просмотра выведет ваш текст (если таковым располагает пользователь). Если нужного шрифта нет, программа проигнорирует запрос и будет использовать шрифт, установленный по умолчанию.

Этот атрибут позволяет указать как один, так и несколько шрифтов (через запятую). Весь список будет просмотрен слева направо и первый из имеющихся на машине пользователя будет использован для вывода документа.

Атрибут `SIZE` служит для указания размера шрифта в условных единицах от 1 до 7. Считается, что размер "нормального" шрифта соответствует числу 3. Размер может быть как абсолютной величиной (`SIZE=5`), так и относительной (`SIZE=+2`). Во втором примере текущий размер шрифта увеличивается на 2.

Атрибут `COLOR` устанавливает цвет шрифта, который может быть задан как в формате RGB, так и указанием имени.

Пример 8.2

Текст `` красного цвета `` и `` большого размера ``



9. Предварительно отформатированный текст

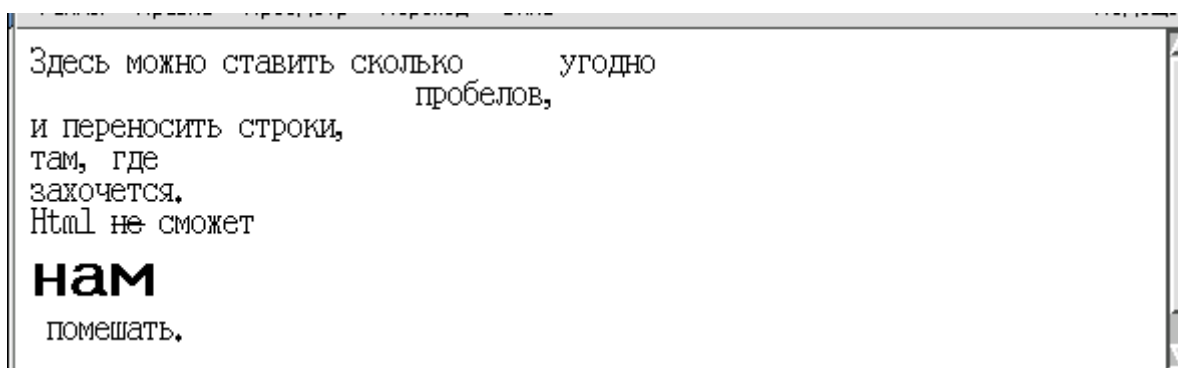
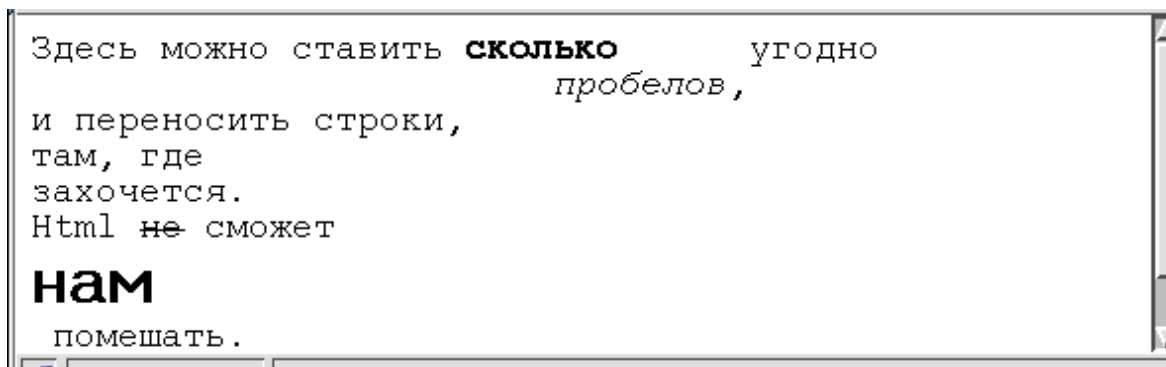
Любой фрагмент текста, расположенный между тегами <PRE> и </PRE>, разбивается на строки и абзацы в точном соответствии с тем, как вы его вводили. Обычно при выводе на экран предформатированного (preformatted) текста используется моноширинный шрифт (так называется шрифт печатной машинки). Этот тег можно использовать, например, для печати стихов, в которых взаимное расположение строк задал сам автор, или для изображения "картинки", состоящей из символов. Другим важным применением тега <PRE> является вывод на экран текстов компьютерных программ (на языках Java, C++ и др.), в которых традиционно используют отступы, дополнительные пробелы и пустые строки для выделения структуры исходного кода. Текст внутри контейнера <PRE> подчиняется действиям тегов <P> и
 и поддерживает теги заголовков. К содержимому контейнера <PRE> могут применяться любые элементы физического и логического форматирования, но некоторые моноширинные шрифты не содержат наборов символов полужирного и курсивного начертания.

Пример 9.1

Рассмотрим фрагмент HTML-документа, содержащий предварительно отформатированный текст:

```
<PRE>
Здесь можно ставить <B>сколько</B>   угодно
                               <EM>пробелов</EM>,
и переносить строки,
там, где
захочется. <P>Html <S>не</S> сможет <H1>нам</H1> помешать.
</PRE>
```

В окне браузера количество пробелов и разрывы строк будут сохранены, к ним добавятся разрывы строк, вызванные использованием тегов <P>,
 и <H1>. Первый из приведенных ниже рисунков соответствует заданию в качестве моноширинного в настройках браузера шрифта Courier (Adobe), а для второго использовался шрифт Fixed (Sony).



10. Мультимедийные возможности HTML

Некоторые браузеры позволяют подключать дополнительные программные модули для мультимедийных приложений. В программу встроена поддержка технологий LiveAudio (для звуковых файлов форматов WAV, AU, AIFF и MIDI), Live3D (VRML), LiveVideo (видео-файлы AVI) и QuickTime (файлы формата MOV, включающие текст и звук MIDI). Все это может быть интегрировано в вашу страницу при помощи тега `<EMBED>`.

Для других браузеров указывается элемент `<NOEMBED>`, где между начальным и конечным тегами указывается альтернативный текст. Отметим некоторые параметры тега `<EMBED>`.

Атрибут	Назначение
<code>SRC</code>	Полный путь или имя файла вставляемого объекта. Параметр обязателен
<code>PALETTE</code>	Тип палитры для просматриваемого объекта. Имеет значения: <code>foreground</code> , <code>background</code>
<code>HIDDEN</code>	Тип отображения объекта на экране. Значения: <code>false</code> -- объект не отображается (по умолчанию), <code>true</code> -- объект отображается
<code>TYPE</code>	Тип файла объекта

Другие атрибуты этого тега аналогичны атрибутам тега `` (собственно, и вставка объекта в HTML-документ происходит похожим образом), поэтому здесь не будут описаны такие параметры элемента, как `NAME`, `ALT`, `BORDER`, `HEIGHT`, `WIDTH`, `ALIGN`, `HSPACE`, `VSPACE`.

Пример

Строка `<EMBED SRC=nature.wav>` позволит использовать в качестве фонового звука музыкальный файл `nature.wav`, а для вложения файла `js.pdf` в формате PDF (Adobe Portable Document Format) нужно добавить строку

```
<EMBED SRC=js.pdf WIDTH=500 BORDER=0 ALIGN=left>
```

Задания 2

1. Создайте файл `first.html`. Напишите свои фамилию и имя, используя заголовок первого уровня.
2. Установите серый цвет фона этого документа.
3. Разместите в документе три пословицы следующим образом: первая выравнивается по левому краю, пишется полужирным шрифтом красного цвета; вторая - по центру курсивным шрифтом зеленого цвета; третья - по правому краю подчеркнутым шрифтом синего цвета.
4. Включите в документ другие образцы логического и физического форматирования текста.
5. Добавьте в документ фрагмент предварительно отформатированного текста. Как ведет себя такой фрагмент при уменьшении ширины окна браузера?

Домашнее задание

1. Создать вторую html-страницу для вашего индивидуального сайта (по выбранной на первом занятии темы).

2. Оформить сайт с применением тегов, рассмотренных на данном занятии.
3. Создать файл PrilA.html. Текст должен соответствовать Приложению А.

Практическое занятие № 6-7 Списки и изображения в HTML

Цель занятия: формировать навык оформления информации в виде списков; работа с графическими изображениями; формирование знаний форматов графических файлов. Включение изображений в HTML.

Часть 1 Списки

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

В настоящее время стандарты HTML поддерживают теги для списков трех различных видов: нумерованных (упорядоченных), маркированных (неупорядоченных) и списков определений. Списки и элементы списков являются блочными элементами. Это означает, что перед ними и после них автоматически добавляются пустые строки.

Язык HTML допускает вложенность любых видов списков. Для этого размещают одну пару тегов (стартовый и завершающий) внутри другой. Следует помнить о том, что все имеющиеся списки должны завершаться закрывающим тегом.

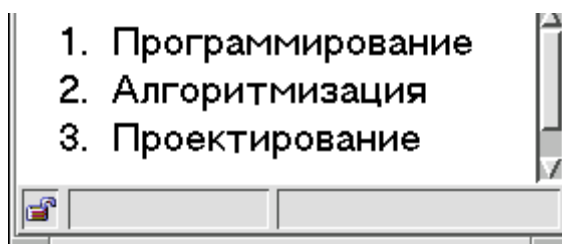
1 Нумерованные списки

Нумерованные (упорядоченные) списки используют, когда важен порядок вывода элементов списка. Браузер автоматически вставляет номера элементов по порядку, в исходном HTML-тексте номера не печатаются. Если количество элементов списка изменится (в результате удаления или добавления новых элементов), то нумерация автоматически обновится.

Весь нумерованный список заключается между парой тегов `` и ``, а каждый элемент списка расположен между тегами `` и `` (закрывающий тег `` может отсутствовать).

Пример 1.1

```
<OL>
<LI>Программирование</LI>
<LI>Алгоритмизация</LI>
<LI>Проектирование</LI>
</OL>
```



Тег `` может имеет атрибуты `TYPE` и `START`:

```
<OL START=n TYPE=вид_счетчика>
```

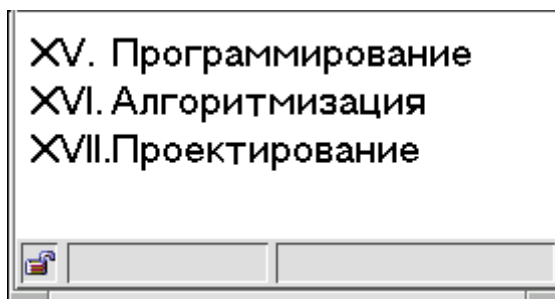
Атрибут `TYPE` задает вид счетчика, возможные значения которого приведены в таблице, а `START` -- начальное значение.

Значение	Функция
A	большие латинские буквы (A,B,C...)
a	маленькие латинские буквы (a,b,c...)
I	большие римские цифры (I,II,III...)

i	маленькие римские цифры (i,ii,iii...)
1	арабские цифры (1,2,3...); используется по умолчанию

Пример 1.2

```
<OL TYPE=I START=15>
<LI>Программирование</LI>
<LI>Алгоритмизация</LI>
<LI>Проектирование</LI>
</OL>
```



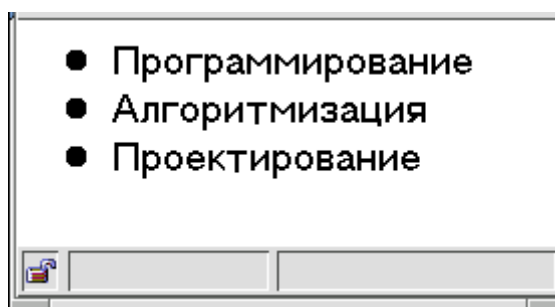
2 Маркированные списки

Маркированный (неупорядоченный) список используется для представления коллекции элементов, порядок вывода которых не важен. При выводе маркированных списков браузер автоматически вставляет специальные значки (маркеры), отмечающие каждый элемент списка.

Маркированный список начинается стартовым тегом `` и завершается тегом ``. Каждый элемент списка начинается с тега `` и завершается (необязательным) тегом ``.

Пример 2.1

```
<UL>
<LI>Программирование</LI>
<LI>Алгоритмизация</LI>
<LI>Проектирование</LI>
</UL>
```



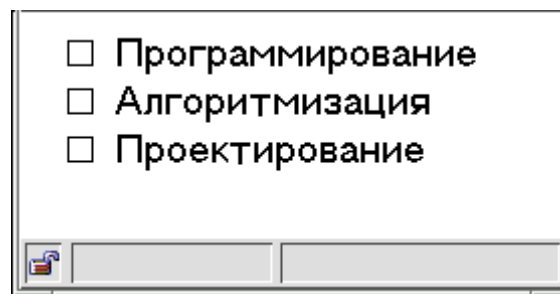
Тег `` имеет атрибут `TYPE`, определяющий внешний вид маркера:

```
<UL TYPE=тип_маркера>
```

Значение атрибута `TYPE` может быть одним из следующих: `disc` (круг -- форма по умолчанию), `circle` (окружность) или `square` (квадрат).

Пример 2.2

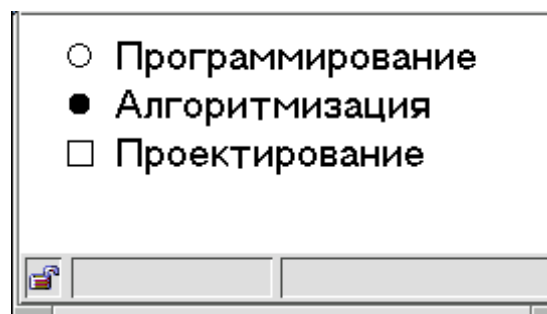
```
<UL TYPE=square>
<LI>Программирование</LI>
<LI>Алгоритмизация</LI>
<LI>Проектирование</LI>
</ul>
```



Атрибут `TYPE` применяется и в теге `` для изменения формы маркера перед конкретным элементом списка.

Пример 2.3

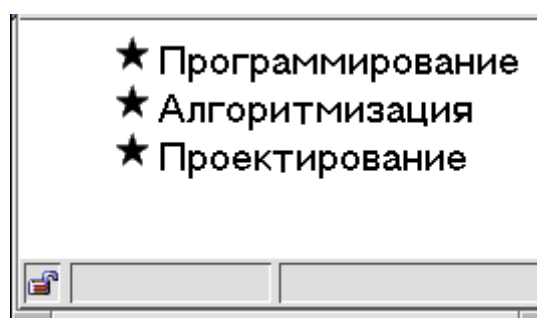
```
<UL>
  <LI TYPE=circle>
    Программирование
  <LI TYPE=disc>
    Алгоритмизация
  <LI TYPE=square>
    Проектирование
</ul>
```



Тег `` обеспечивает вывод маркера и разделение элементов списка. Если хочется использовать нестандартные маркеры, то тег `` не указывается. Для выделения элементов списка в этом случае используются какие-либо картинки или символы, а тег `
` обеспечивает переход к следующему элементу списка.

Пример 2.4

```
<UL>
  <IMG SRC=arr.gif ALIGN=top>
  <nbr>Программирование<BR>
  <IMG SRC=arr.gif ALIGN=top>
  <nbr>Алгоритмизация<BR>
  <IMG SRC=arr.gif ALIGN=top>
  <nbr>Проектирование<BR>
</UL>
```



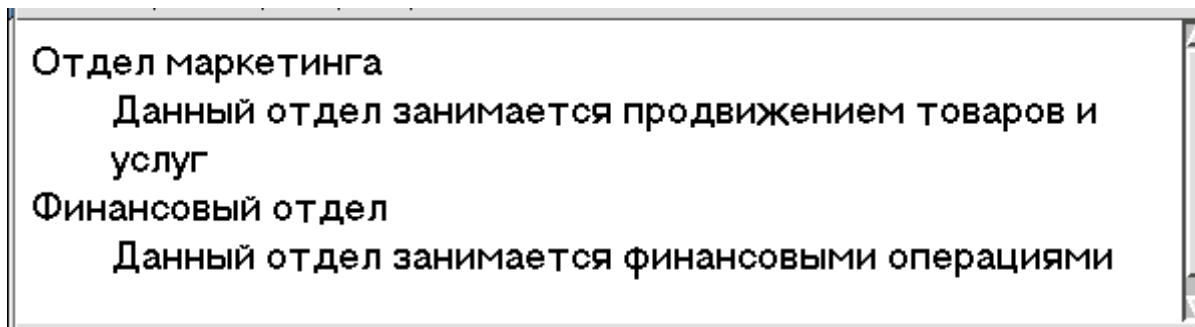
3 Список определений

Список описаний (список определений) начинается с тега `<DL>` и завершается тегом `</DL>`. Данный список служит для создания списков типа «термин» – «описание». Термин автоматически размещается у левой границы страницы, а их определения смещены относительно них вправо. Каждый термин обозначается тегом `<DT>`, а его описание – тегом `<DD>`.

Пример 3.1

```
<DL>
  <DT>Отдел маркетинга
  <DD>Данный отдел занимается продвижением товаров и услуг
  <DT>Финансовый отдел
  <DD>Данный отдел занимается финансовыми операциями
</DL>
```

В результате браузер покажет следующее:

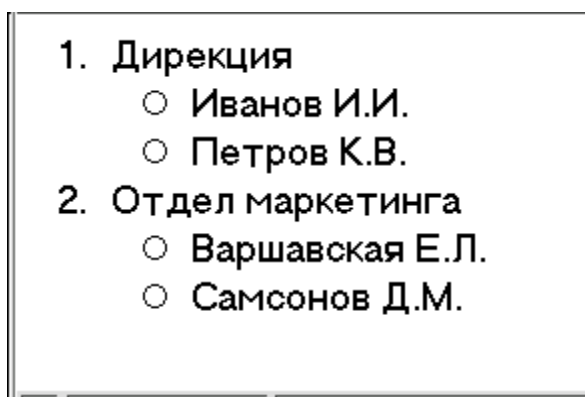


Отметим, что все вышеперечисленные виды списков имеют атрибут **COMPACT**, позволяющий выводить списки в более компактном виде.

4 Вложенные списки

Любой список может быть частью другого списка, вложен в другой список. Считается полезным использование сдвигов при подготовке текста исходного HTML-документа, чтобы четко представлять уровни вложенности списков.

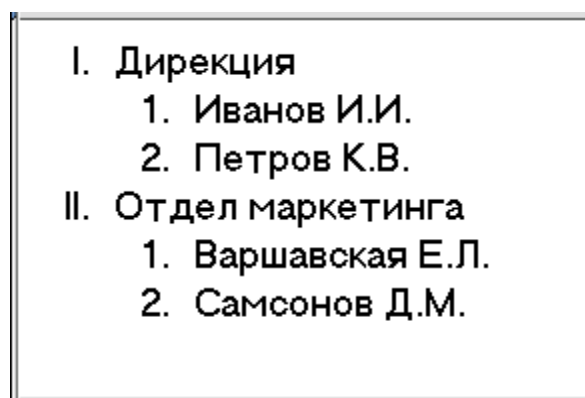
```
<OL>
<LI>Дирекция</LI>
<UL>
<LI>Иванов И.И.</LI>
<LI>Петров К.В.</LI>
</UL>
<LI>Отдел маркетинга</LI>
<UL>
<LI>Варшавская Е.Л.</LI>
<LI>Самсонов Д.М.</LI>
</UL>
</OL>
```



При выводе вложенных маркированных списков браузер автоматически проставляет маркеры перед элементами, находящимися на разных уровнях вложенности.

Для вложенных нумерованных списков браузер, к сожалению, не изменяет тип нумерации. По умолчанию каждый уровень внутри такого списка будет отмечен арабскими цифрами. Для получения списка в другом формате потребуется вручную проставить каждую метку, используя атрибут **TYPE**.

```
<OL TYPE=I>
<LI>Дирекция</LI>
<OL>
<LI>Иванов И.И.</LI>
<LI>Петров К.В.</LI>
</OL>
<LI>Отдел маркетинга</LI>
<OL>
<LI>Варшавская Е.Л.</LI>
<LI>Самсонов Д.М.</LI>
```



Задания

1. Создайте файл `list.html` в директории учебного сайта и включите в него маркированные и нумерованные списки нескольких уровней вложенности.
2. Используйте различные стартовые значения для нумерованных списков.
3. Создать файл `PrilB.html`. Текст должен соответствовать [Приложению Б](#).

Часть 2 Изображения в HTML

Рисунки и анимация могут сделать HTML-документ более привлекательным и интересным. Они не только украшают страницу, но и помогают лучше передать содержание документа. Для правильного использования графики в HTML-документе необходимо учитывать следующие факторы: многие браузеры поддерживают только графические форматы GIF и JPEG; файлы, содержащие графику, передаются медленно; некоторые пользователи не имеют графических браузеров или намерено отключают загрузку изображений; цветная графика, которая хорошо смотрится на вашем компьютере, может плохо выглядеть на другом.

Тег `` вставляет изображение в документ, как если бы оно было просто одним большим символом. Синтаксис тега:

``

Атрибуты тега `` и их значения приведены в таблице.

Атрибут	Назначение
<code>SRC="файл"</code>	Задаёт URL-адрес изображения (можно указывать как абсолютный, так и относительный URL-адрес; если файл с изображением находится в той же директории, что и HTML-документ, то достаточно просто указать имя файла); этот атрибут является обязательным
<code>ALT="текст"</code>	Задаёт альтернативный текст для браузеров, не поддерживающих работу с изображениями
<code>ALIGN="тип"</code>	Задаёт расположение картинки относительно текста, тип может принимать следующие значения: TOP, MIDDLE, BOTTOM, LEFT, RIGHT
<code>BORDER=n</code>	Устанавливает толщину обрамления вокруг изображения в пикселах
<code>HEIGHT=n(%)</code>	Устанавливает высоту изображения в пикселах или в процентах от высоты окна браузера
<code>WIDTH=n(%)</code>	Устанавливает ширину изображения в пикселах или в процентах
<code>HSPACE=n</code>	Задаёт свободное пространство слева и справа от изображения (в пикселах)
<code>VSPACE=n</code>	Задаётся свободное пространство над и под изображением (в пикселах)

Обратите внимание, что ширина и высота изображения могут быть заданы не только в пикселах, но и в процентах от размеров окна браузера. Многие компоненты, включаемые в состав Web-страниц (изображения, таблицы, апплеты и т. д.), позволяют задавать размер в относительных единицах (т. е. в процентах). Это позволяет уменьшить зависимость внешнего вида документа от текущих установок конкретного браузера и особенностей операционной системы. Рекомендуется задавать только один из атрибутов пары "ширина-высота" изображения, иначе рисунок может быть непропорционально деформирован и изменит свой вид.

Пример 1

Если размер изображения, хранящегося в файле `exm2.gif` составляет 150 пикселей по ширине и 90 по высоте, то следующая команда приведет к включению в документ деформированного изображения: `` .

Помните, что графика передается по сети намного медленнее, чем текст, поэтому лучше включать в документ картинки небольшого размера. Если вы задаете размер изображения меньше, чем у исходного, то это приводит только к неоправданному увеличению объема передачи информации по сети. Рациональнее предварительно уменьшить размер изображения с помощью графического редактора.

Язык HTML позволяет задать расположение изображения относительно окружающего его текста. Атрибут `ALIGN` может принимать следующие значения.

Значение	Функция
<code>TOP</code>	Выравнивает одну строку по верху изображения, остальные помещает после рисунка
<code>MIDDLE</code>	Выравнивает одну строку по середине изображения, остальные помещает после рисунка
<code>BOTTOM</code>	Выравнивает одну строку по низу изображения, остальные помещает после рисунка
<code>LEFT</code>	Прижимает обтекаемое текстом изображение к левой стороне окна браузера
<code>RIGHT</code>	Прижимает обтекаемое текстом изображение к правой стороне окна браузера

Пример 2

Рассмотрим возможные варианты выравнивания изображения относительно текста.

``



Первая строка текста находится сверху изображения и обычно используется для описания рисунка.

``



Первая строка текста находится по середине изображения и используется для того же.

``



Первая строка текста находится внизу изображения и используется так же, как и в двух первых случаях.

``



Изображение прижимается к левому краю окна просмотра, а текст обтекает изображение справа.

``

Изображение прижимается к правому краю окна просмотра, а текст обтекает изображение слева.



Задания

1. Запустите браузер, выберите любую картинку и скопируйте ее к себе в директорию.
2. В документ, хранящийся в файле `image.html`, включите скопированное изображение, при этом рисунок должен иметь рамку шириной 3 px и прижиматься к правому краю, а расстояние вокруг рисунка должно составлять 5 px. Включите альтернативный текст в описание тега `IMG`.

Самостоятельная работа

1. Разработать техническое задание для создания сайта по индивидуальной теме.
2. Создать страницы сайта по индивидуальной теме с применением изученных тегов.

Практическое занятие № 8-9 Ссылки в HTML

Цель занятия: формировать навык создания гиперссылок, организации навигации на сайте, создания навигационного меню; знания различных типов меню, создания карты-изображения.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Несмотря на то, что в состав HTML-документа входят самые различные компоненты, можно сказать, что **гипертекстовые ссылки** - основа WWW. Если бы Web-страницы не ссылались друг на друга, содержимое Web превратилось бы в обычный набор файлов, не связанных между собой.

Для создания гипертекстовой ссылки используется пара тегов `<A>...`. Фрагмент текста, изображение или любой другой объект, расположенный между этими тегами, отображается в окне браузера как гипертекстовая ссылка. Активация такого объекта приводит к загрузке в окно браузера нового документа или к отображению другой части текущей Web-страницы. Гипертекстовая ссылка формируется с помощью выражения

`фрагмент документа`

`HREF` здесь является обязательным атрибутом, значение которого и есть URL-адрес запрашиваемого ресурса. Кавычки в задании значения атрибута `HREF` не обязательны.

Текстовые указатели, т. е. фрагменты текста, являющиеся ссылками, не отличаются разнообразием внешнего вида. Обычно такой указатель представляет собой слово или слова, подчеркнутые прямой линией. Цвет указателя может регулироваться автором и установками программы просмотра.

Пример 1

`Ссылка`

Графические указатели, т. е. изображения-ссылки, выделяются рамкой того же цвета, что и текстовые указатели.

Пример 2

``

1 Абсолютный и относительный URL

В разделе, посвященном компьютерным сетям и принципам поиска информации в Интернет, было введено понятие URL (универсального локатора ресурса), однозначно определяющего расположение объекта в глобальной сети. В общем случае URL имеет следующий вид (части, заключенные в квадратные скобки, не обязательны и могут быть опущены):

протокол://адрес_узла[:порт]/путь/файл[#метка]

В таблице ниже приведены назначения компонент URL.

Компонента	Назначение
протокол	Обозначение одного из протоколов, используемых для обращения к ресурсу, возможные значения: http, ftp, file и др.
адрес_узла	Доменное имя или IP-адрес компьютера в сети Интернет
порт	Порт, по которому клиент обращается к серверу для установления соединения; указывается только в случае обращения к нестандартному порту
путь	Путь к требуемому ресурсу
файл	Имя файла, содержащего HTML-документ или другой ресурс
метка	Позиция в документе, начиная с которой он отображается в окне браузера

URL, заданный в таком виде, называется **абсолютным URL**, так как он полностью описывает расположение ресурса в глобальной сети.

Пример 1.1

При обращении к конкретному ресурсу порт и позиция в документе часто не указываются:

<http://www.ctc.msiu.ru/education/book/index1.html>

Пример 1.2

Рассмотрим ссылку на так называемую *домашнюю страницу* (Home Page) конкретного пользователя. Предположим, что на сервере www.msiu.ru зарегистрирован человек с пользовательским именем (login) [ivanov](#). Домашней страницей называют файл [index.html](#), находящийся в директории с именем [public_html](#), которая, в свою очередь, должна располагаться в домашней директории пользователя. Тогда ссылка на домашнюю страницу этого человека может быть задана в виде

`Текст ссылки`

Обратите внимание, что имя самого файла при такой записи не указывается. Если потребуется сослаться на какой-либо другой документ данного пользователя, расположенный в директории `public_html`, например, `photo.html`, то ссылка примет вид:

```
<A HREF="http://www.msiu.ru/~ivanov/photo.html">Текст</A>
```

Относительный URL описывает положение ресурса, на который указывает ссылка, относительно URL текущего документа.

Пример 1.3

При задании относительных ссылок указывается путь по файловому дереву до того места, где находится требуемый HTML-ресурс. Вот ссылка на документ `image.html`, размещенный в текущем каталоге: `Ссылка`.

Если файл `pict.html` лежит в родительском по отношению к текущему документу каталоге, то следует использовать запись

```
<A HREF="../pict.html">Текст ссылки</a>
```

Если же требуемый документ находится в поддиректории `Picture`, то используется запись, аналогичная следующей

```
<A HREF="Picture/pict.html">Текст ссылки</A>
```

2 Гиперссылки в пределах одного документа

Другая форма тега `<A>` предназначена для присваивания имени некоторому фрагменту документа HTML:

```
<A NAME="имя">фрагмент документа</A>
```

Тег `<A>` часто называют тегом *якоря* (anchor). Если якорь применяется для добавления метки (имени) к фрагменту документа, то его называют *именованным якорем*.

Для того чтобы сослаться на фрагмент, которому присвоено имя, используется следующая форма тега `<A>`:

```
<A HREF="URL_ресурса#имя">Текст ссылки</A>
```

Если нужно сослаться на фрагмент текущего документа, то URL-ресурса можно опустить:

```
<A HREF="#имя">Текст ссылки</A>
```

Пример 2.1

Пусть в документе с именем `book.html` заголовку главы 2 присвоено имя `chapter2`:

```
<A NAME="chapter2">Глава 2</A>
```

Тогда ссылка на эту главу, расположенная в этом же документе, будет иметь вид:

```
<A HREF="#chapter2">Глава 2</A>
```

Для задания ссылки на эту же метку файла `book.html` из другого файла нужно написать:

`Глава 2`

3 Ссылка на почтовый ящик

Для того чтобы создать ссылку на почтовый ящик, напишите:

`текст ссылки`

Здесь вместо `адрес_e-mail` нужно поставить адрес почтового ящика. После того, как посетитель активизирует эту ссылку, у него на экране появится специальное окно, позволяющее послать письмо по указанному адресу.

Пример 3.1

`Почта автору`

Задания

1. Откройте файл `image.html` и сделайте в нем ссылку в виде картинки на файл `first.html`.
2. Откройте файл `first.html` и сделайте в нем текстовую ссылку на файл `image.html`.
3. Добавьте в файл `first.html` несколько абзацев текста в таком количестве, чтобы документ уже не помещался целиком в окне браузера. Отметьте последний абзац, присвоив ему имя `end`.
4. Включите ссылку, ведущую сразу к последнему абзацу файла `first.html`, в файл `image.html`.
5. В файл `first.html` включите ссылку на свой почтовый ящик.
6. Создайте карту-изображение, используя одно фото с несколькими объектами.

Практическое занятие №10 Таблицы

Цель занятия: формировать навык по разработке таблиц средствами HTML-разметки.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

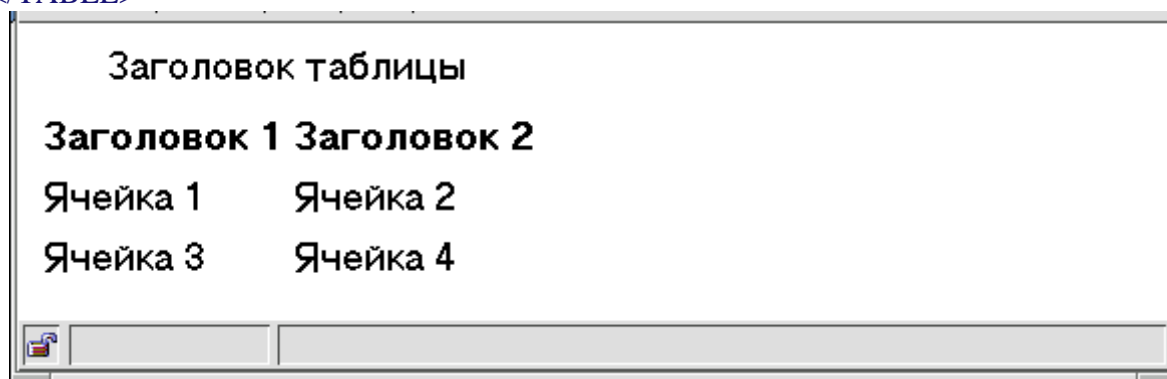
Теги HTML для создания таблиц первоначально предназначались для представления строк и столбцов табулированных данных. Однако дизайнеры научились с их помощью управлять разметкой Web-страниц: создавать столбцы текста, задавать интервалы между элементами и изменять внешний вид текста способами, недоступными другим тегам форматирования HTML.

Таблицы в языке HTML всегда имеют прямоугольный вид и состоят из строк, которые в свою очередь состоят из ячеек. Все языковые конструкции, описывающие компоненты создаваемой таблицы, заключаются между тегами `<TABLE>` и `</TABLE>`.

Заполнение таблицы происходит построчно; для обозначения строки используется пара тегов `<TR>...</TR>`. Строка состоит из ячеек, для задания которых используют либо теги `<TH>...</TH>`, если эти ячейки содержат заголовки столбцов, либо теги `<TD>...</TD>`. Заголовки выводятся полужирным шрифтом и располагаются по центру ячейки. Данные имеют обычный шрифт и выравниваются по левой стороне ячейки. Для задания заголовка всей таблицы используются теги `<CAPTION>` и `</CAPTION>`.

Пример 1

```
<TABLE>
<CAPTION>Заголовок таблицы</CAPTION>
<TR><TH>Заголовок 1</TH><TH>Заголовок 2</TH></TR>
<TR><TD>Ячейка 1</TD><TD>Ячейка 2</TD></TR>
<TR><TD>Ячейка 3</TD><TD>Ячейка 4</TD></TR>
</TABLE>
```



Заголовок 1	Заголовок 2
Ячейка 1	Ячейка 2
Ячейка 3	Ячейка 4

Наличие в ячейках данных не обязательно. Создать пустую ячейку можно двумя способами: ничем не заполнять ее контейнер (`<TD></TD>`), либо поместить в нее символ неразрывного пробела, задаваемого специальной последовательностью символов -- ` `; (т. е. создать ячейку вида `<TD> </TD>`).

Нет надобности отдельно создавать пустые ячейки, если планируется, что все оставшиеся в строке ячейки не будут заполнены. Так как тег `<TR>` сигнализирует о начале новой строки, пустые ячейки будут добавлены браузером автоматически.

Основные атрибуты тега <TABLE>

Назначение основных атрибутов тега <TABLE> и значения, которые они могут принимать перечислены в таблице.

Атрибут	Назначение
<code>BORDER=n</code>	Определяет ширину рамки таблицы (в пикселах), например, <code>BORDER=1</code> ; значение, равное нулю, означает отсутствие рамки
<code>WIDTH=n</code>	Определяет ширину всей таблицы в пикселах, либо в процентах от ширины окна браузера
<code>HEIGHT=n</code>	Определяет высоту всей таблицы в пикселах, либо в процентах от высоты окна браузера
<code>ALIGN</code>	Задаёт горизонтальное выравнивание таблицы в окне браузера; имеет значения <code>left</code> , <code>center</code> и <code>right</code> (по умолчанию -- <code>left</code>)
<code>CELLPADDING=n</code>	Добавляет свободное пространство между данными внутри ячейки и ее границами; по умолчанию значение равно 2
<code>CELLSPACING=n</code>	Добавляет свободное пространство между ячейками внутри всей таблицы; по умолчанию значение равно 2
<code>HSPACE=n</code>	Задаёт области свободного пространства указанной ширины (в пикселах) слева и справа от таблицы
<code>VSPACE=n</code>	Задаёт области свободного пространства заданной высоты (в пикселах) сверху и снизу от таблицы
<code>BGCOLOR=цвет</code>	Устанавливает цвет фона всей таблицы
<code>BACKGROUND=URL</code>	Указывает фоновое изображение для таблицы, где URL -- адрес источника (имя файла с изображением)

Пример 2

Изменим содержимое документа, созданного в предыдущем примере, добавив атрибуты `BORDER` и `ALIGN` в тег <TABLE>:

```
<TABLE BORDER="1" ALIGN="center">
```

Теперь ячейки таблицы будут обрамлены рамкой, а таблица выровнена по центру окна браузера.

Заголовок таблицы	
Заголовок 1	Заголовок 2
Ячейка 1	Ячейка 2
Ячейка 3	Ячейка 4

Выравнивание данных в ячейках

При помощи атрибутов **ALIGN** и **VALIGN** можно по-разному размещать данные относительно границ ячейки. Эти атрибуты используются совместно с тегами **<CAPTION>**, **<TR>**, **<TH>** и **<TD>** в самых различных комбинациях. Ниже приведены значения атрибутов для перечисленных элементов.

Тег	Назначение атрибута
<TR>	<p>Атрибут ALIGN может принимать значения left, center и right (по умолчанию -- left для данных и center для заголовков); он определяет горизонтальное выравнивание данных в ячейках и действует на всю строку, если не отменяется тем же атрибутом в отдельной ячейке</p> <p>Атрибут VALIGN может иметь значения top, bottom, middle и baseline (по умолчанию -- middle); он регулирует положение данных относительно верхней и нижней границ ячейки и влияет на всю строку, если не отменяется таким же атрибутом в отдельной ячейке. baseline применяется ко всем элементам строки и выравнивает их по базовой линии</p>
<TH>	<p>Атрибут ALIGN может принимать значения left, center и right (по умолчанию -- center)</p> <p>Атрибут VALIGN может иметь значения top, bottom и middle (по умолчанию -- middle)</p>
<TD>	<p>Атрибут ALIGN может принимать значения left, center и right (по умолчанию -- left)</p> <p>Атрибут VALIGN может иметь значения top, bottom и middle (по умолчанию -- middle)</p>
<CAPTION>	<p>Атрибут ALIGN может иметь значения top и bottom (по умолчанию -- top); размещает заголовок таблицы сверху или снизу</p>

Объединение ячеек

Смежные ячейки таблицы могут объединяться. Например, в таблице из нескольких столбцов все ячейки первой строки можно объединить и поместить в этой строке красивый заголовок таблицы. Возможно также объединение нескольких строк или создание пустой прямоугольной области.

Для соединения двух смежных ячеек в одном столбце нужно использовать атрибут **ROWSPAN** тега `<TH>` или `<TD>`, например,

```
<TD ROWSPAN=2>
```

Для объединения двух смежных ячеек в одной строке нужно использовать атрибут **COLSPAN** тех же тегов, например,

```
<TD COLSPAN=2>
```

Пример 3

В следующей таблице используется объединение столбцов и строк.

```
<HTML>
<BODY>
<TABLE BORDER=1 ALIGN=center>
<TR><TH COLSPAN=3>Заголовок на 3 столбца</TH></TR>
<TR>
<TH>Заголовок на 1 строку</TH>
<TD>Ячейка 1</TD>
<TD>Ячейка 2</TD>
</TR>
<TR>
<TH ROWSPAN=3>Заголовок на 3 строки</TH>
<TD>Ячейка 3</TD>
<TD>Ячейка 4</TD>
</TR>
<TR><TD>Ячейка 5</TD><TD>Ячейка 6</TD></TR>
<TR><TD>Ячейка 7</TD><TD>Ячейка 8</TD></TR>
</TABLE>
</BODY>
</HTML>
```

Таблица будет иметь следующий вид.

Заголовок таблицы		
Заголовок на 3 столбца		
Заголовок на 1 строку	Ячейка 1	Ячейка 2
Заголовок на 3 строки	Ячейка 3	Ячейка 4
	Ячейка 5	Ячейка 6
	Ячейка 7	Ячейка 8

Если вы хотите создать таблицу с объединениями столбцов и в то же время точно контролировать ширину каждого столбца, необходимо задать ширину по крайней мере одной ячейки в каждом столбце. Для полной уверенности найдите время и задайте ширину каждой

ячейки в таблице. Когда объединения столбцов пересекаются, очень легко получить непредсказуемый результат.

Цвет в таблицах

В HTML не предусмотрено специальных средств раскрашивания таблиц. Однако некоторые браузеры имеют расширения, позволяющие изменять цвет ячеек и рамок. Вы можете изменить цвет фона ячейки при помощи атрибута **BGCOLOR** перед размещением в ней текста или изображения, а также использовать атрибут **BORDERCOLOR** для изменения цвета рамки ячейки. Теги **<TABLE>**, **<TD>**, **<TH>** и **<TR>** также допускают использование в них указанных атрибутов. Таким образом, вы можете изменить цвет всей таблицы, отдельной ячейки или строки таблицы.

Значения цветов, установленные на уровне ячейки, будут перекрывать значения, установленные на уровне строки, которые в свою очередь, будут перекрывать значения, заданные на уровне всей таблицы.

Пример 4

Создадим таблицу, удовлетворяющую следующим требованиям:

- цвет таблицы -- белый;
- вторая строка светло-серого цвета, в ячейке 2 -- цвет "teal";
- цвет надписи в ячейке 5 -- красный;
- первый столбец составляет 20% от ширины таблицы, два другие -- по 40%;
- ячейка 3 центрирована, а ячейка 4 -- выровнена вправо;
- ячейки 6 и 8 объединены в одну, центрированы и выровнены по нижнему краю ячейки;
- поля внутри ячеек -- 10 пикселей.

Простой заголовок	Заголовок на 2 столбца	
Заголовок на 1 строку	Ячейка 1	Ячейка 2
Заголовок на 3 строки	Ячейка 3	Ячейка 4
	Ячейка 5	Ячейка 6
	Ячейка 7	

<HTML>

<BODY>

<TABLE BORDER=8 WIDTH=90% BGCOLOR="white"

```

CELLPADDING=10 ALIGN=center>
<TR>
<TH WIDTH=20%>Простой заголовок</TH>
<TH WIDTH=80% COLSPAN=2>Заголовок на 2 столбца</TH>
</TR>
<TR BGCOLOR="#CCCCCC">
<TH WIDTH=20%>Заголовок на 1 строку</TH>
<TD WIDTH=40%>Ячейка 1</TD>
<TD WIDTH=40% BGCOLOR="teal">Ячейка 2</TD>
</TR>
<TR>
<TH WIDTH=20% ROWSPAN=3>Заголовок на 3 строки</TH>
<TD ALIGN=center>Ячейка 3</TD>
<TD ALIGN=right>Ячейка 4</TD>
</TR>
<TR>
<TD><FONT COLOR="red">Ячейка 5</FONT></TD>
<TD ROWSPAN=2 ALIGN=center VALIGN=bottom>Ячейка 6</TD>
</TR>
<TR>
<TD>Ячейка 7</TD>
</TR>
</TABLE>
</BODY>
</HTML>

```

Задание

1. Создайте файл `table.html`, в котором разместите приведенную выше таблицу, оформите макеты приведенных ниже трех таблиц.

Таблица 1.

Таблица 2.

Таблица 3.

2. Оформить в виде таблицы Статью из книги (Приложение В).

Практическое занятие № 11 – 12 Формы в HTML-документах

Цель занятия: формировать навык по разработке сайта с использованием форм.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Формы HTML позволяют получать информацию от читателей. До сих пор мы обсуждали только способы вывода данных, теперь речь пойдет об обратном действии. Формы дают возможность запрашивать информацию в виде свободного текста, получать ответы типа "да/нет" или делать выбор из нескольких опций.

Вы можете использовать формы с различными целями. Простейшим примером является размещение формы, куда читатели, посетившие сайт, смогут записать свои отзывы. Круг применения форм HTML ограничивается только вашей фантазией.

1 Тег <FORM>

Этим тегом начинается каждая форма. В нем нужно определить два атрибута, указывающих используемый скрипт и метод отправки данных.

Атрибут	Назначение
ACTION	Определяет URL, который примет и обработает данные формы. Если этот атрибут не определен, данные отправляются по адресу страницы, на которой помещена форма
METHOD	Указывает форме, как послать информацию соответствующей программе обработки (скрипту). Обычно он получает значение <code>post</code> , тогда информация формы посылается отдельно от URL. Значению <code>get</code> соответствует посылка вместе с URL

Пример

```
<FORM METHOD="post" ACTION="/cgi-bin/comment_script">  
...  
</FORM>
```

В этом примере дано указание браузеру отправить заполненную форму для обработки скриптом `comment_script`, расположенным в каталоге `cgi-bin` вашего сервера, и использовать метод отправки `post`.

На странице можно расположить любое число форм, однако, нужно следить за тем, чтобы не поместить одну форму в другую.

2 Работа с тегами форм

В HTML существует три тега для создания различного типа полей в форме: `<TEXTAREA>`, `<SELECT>` и `<INPUT>`. Любое их количество может быть размещено в контейнере между тегами `<FORM>` и `</FORM>`. Ниже дано их краткое описание, а подробнее они будут рассмотрены чуть позже.

Тег	Назначение
-----	------------

<code><TEXTAREA></code>	Определяет поле, в которое пользователь вводит многострочную текстовую информацию
<code><SELECT></code>	Позволяет пользователю сделать выбор в окне с полосой прокрутки, либо в раскрывающемся меню
<code><INPUT></code>	Обеспечивает некоторые другие виды ввода информации: ввод одной строки текста, установку и сброс флажков (check boxes), выбор переключателя (radio buttons) и нажатие кнопки для отправки данных или очистки формы

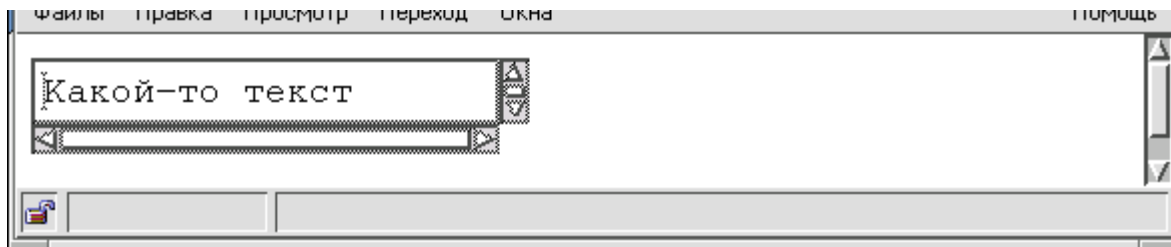
3 Тег `<TEXTAREA>`

Этот тег предназначен для построения поля с целью ввода многострочный текстовой информации. В контейнере `TEXTAREA` допускается размещать любой текст, который будет выведен в поле ввода по умолчанию. Перечислим атрибуты этого тега.

Атрибут	Назначение
<code>NAME</code>	Определяет название поля; обязателен
<code>ROWS</code>	Устанавливает высоту поля, т. е. число строк в нем
<code>COLS</code>	Устанавливает ширину поля, т. е. длину строки

Пример

```
<HTML><BODY>
<FORM>
<TEXTAREA> Какой-то текст </TEXTAREA>
</FORM>
</BODY></HTML>
```



При помощи атрибутов `ROWS` и `COLS` можно задать поле любого размера. Хотя эти атрибуты не являются обязательными, они не имеют определенных значений по умолчанию (для каждого браузера эти значения различны), поэтому лучше их всегда указывать явно.

4 Тег `<SELECT>`

Этот тег используется для создания всплывающего меню или списка опций с полосой прокрутки. Список опций и пункты меню располагаются внутри контейнера `SELECT`.

Аналогично тегу `<TEXTAREA>`, `<SELECT>` требует обязательного определения имени в атрибуте `NAME`. Количество опций указывается в атрибуте `SIZE`. Ниже перечислены атрибуты тега `<SELECT>`.

Атрибут	Назначение
<code>NAME</code>	Определяет название информации
<code>SIZE</code>	Определяет вертикальный размер окна для опций выбора. Если атрибут опущен или его значение равно 1, выводится всплывающий список опций. Если указано число больше единицы, то опции выводятся в окне с полосой прокрутки. Если значение атрибута больше, чем фактическое количество элементов списка, добавляются пустые строки. При их выборе пользователем возвращаются пустые поля
<code>MULTIPLE</code>	Позволяет выбирать сразу нескольких опций

Список опций включается в контейнер `<SELECT>` при помощи тега `<OPTION>`. Этот тег имеет два атрибута.

Атрибут	Назначение
<code>VALUE</code>	Указывает значение, возвращаемое программе обработки (скрипту), в случае выбора опции пользователем
<code>SELECTED</code>	Указывает на опцию, выбранную по умолчанию

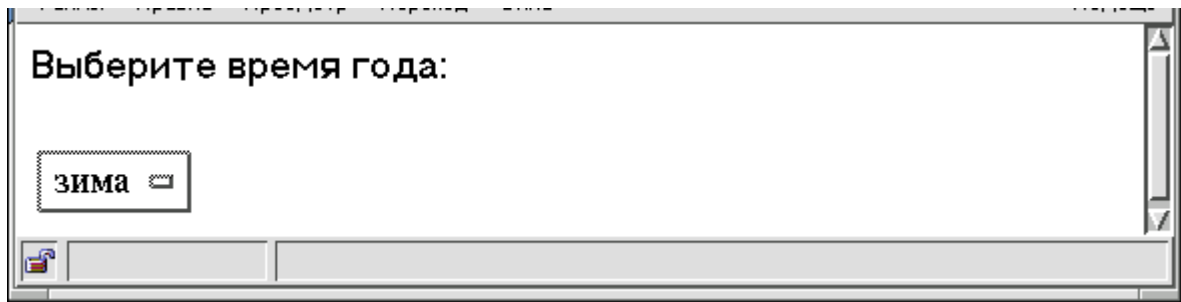
Пример

Ниже приведено содержимое двух HTML-документов, содержащих похожие формы. Во втором документе предварительно выбран один элемент из списка (добавлен атрибут `SELECTED`).

```

<HTML>
<BODY>
  Выберите время года:
  <FORM>
  <SELECT NAME=year>
  <OPTION SELECTED VALUE="winter"> зима
  <OPTION VALUE="spring"> весна
  <OPTION VALUE="summer"> лето
  <OPTION VALUE="autumn"> осень
  </SELECT>
  </FORM>
</BODY>
</HTML>

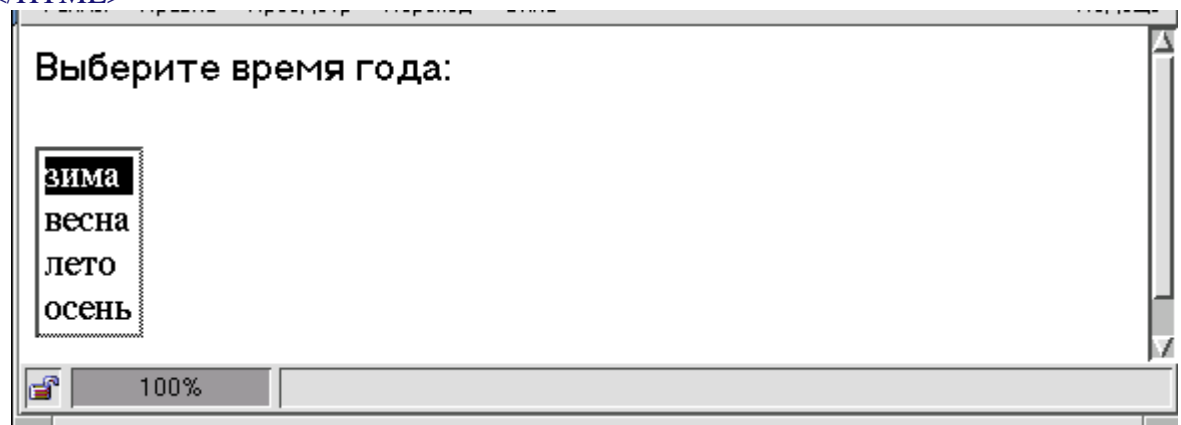
```



```

<HTML>
<BODY>
<FORM>
  Выберите время года:
  <SELECT MULTIPLE NAME="year">
  <OPTION SELECTED VALUE="winter"> зима
  <OPTION VALUE="spring"> весна
  <OPTION VALUE="summer"> лето
  <OPTION VALUE="autumn"> осень
  </SELECT>
</FORM>
</BODY>
</HTML>

```



5 Тег <INPUT>

Тег <INPUT>, в отличие от <TEXTAREA> и <SELECT>, является одиночным тегом. Он предназначен для сбора информации различными способами, включая текстовые поля, поля для ввода пароля, переключатели, флажки, кнопки для отправки данных (Submit) и для очистки формы (Reset, Clear).

Тег <INPUT> располагает следующими атрибутами.

Атрибут	Назначение
NAME SIZE	Указывает размер поля ввода в символах
MAXLENGTH	Определяет максимально возможное число символов, вводимых в поле
VALUE	Для текстового поля определяет текст, выводимый по умолчанию. Для флажков и переключателей указывает значение, возвращаемое программе обработки. Для кнопок отправки и очистки формы определяет надпись на кнопке

CHECKED	Устанавливает флажок или переключатель во включенное состояние по умолчанию. С другими типами тегов <code><INPUT></code> не употребляется
TYPE	Устанавливает тип поля ввода

6 Тип поля ввода, атрибут **TYPE**

Атрибут **TYPE** тега `<INPUT>` может принимать следующие значения.

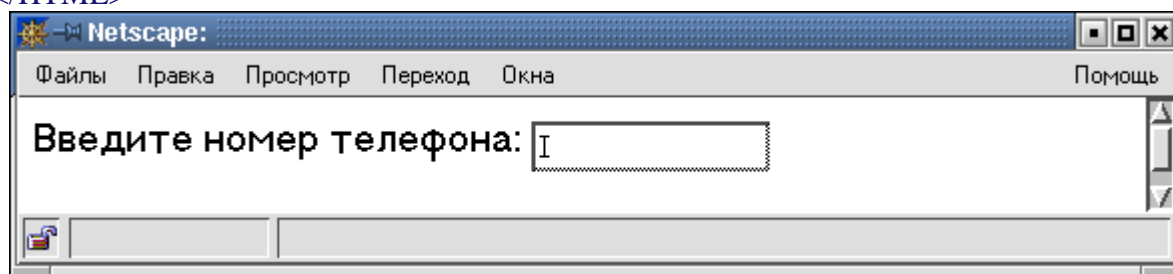
Атрибут	Назначение
TEXT	Является значением по умолчанию и предполагает создание одной строки для ввода данных. Для этого типа поля ввода употребляются атрибуты NAME (обязательный), SIZE , VALUE и MAXLENGTH
PASSWORD	Позволяет заменять вводимые символы пароля звездочками. Для этого типа поля ввода используются атрибуты NAME (обязательный), SIZE , MAXLENGTH и VALUE
CHECKBOX	Позволяет вывести поле для установки флажка в виде маленького квадратика, в котором может быть произведена отметка опции "галочкой". Может использоваться совместно с атрибутами NAME (обязательный), VALUE и CHECKED (определяет установленный по умолчанию флажок). Флажки обычно употребляются, когда можно выбрать сразу несколько опций из числа предложенных. Нужно быть очень осторожным в использовании флажков и переключателей, если цвет фона страницы определяется не документом, а пользователем при помощи установок программы просмотра. Не допускайте, чтобы опции сливались с фоном страницы
RADIO	Позволяет выбрать только одну из представленного числа опций. Переключатели можно группировать, задавая одно и то же значение атрибута NAME (обязательный). Так же используются атрибуты VALUE и CHECKED
RESET	Позволяет создать кнопку для очистки формы. Атрибут VALUE может быть использован здесь для наименования этой кнопки (по умолчанию кнопка имеет надпись Reset)
SUBMIT	Используется для создания кнопки, по нажатию которой введенные данные отправляются на сервер для обработки программой-скриптом. В атрибуте VALUE может быть указано название для этой кнопки (по умолчанию -- Submit Query)

Пример

В следующей форме используется значение **TEXT**.

```
<HTML>
<BODY>
<FORM>
Введите номер телефона:
<INPUT TYPE="TEXT" NAME="phone"
SIZE="15" MAXLENGTH="12">
</FORM>
```

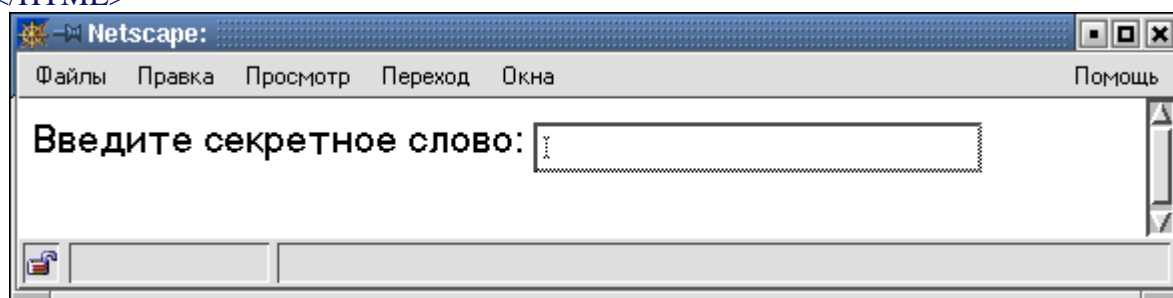
```
</BODY>
</HTML>
```



Пример

Использование значения PASSWORD.

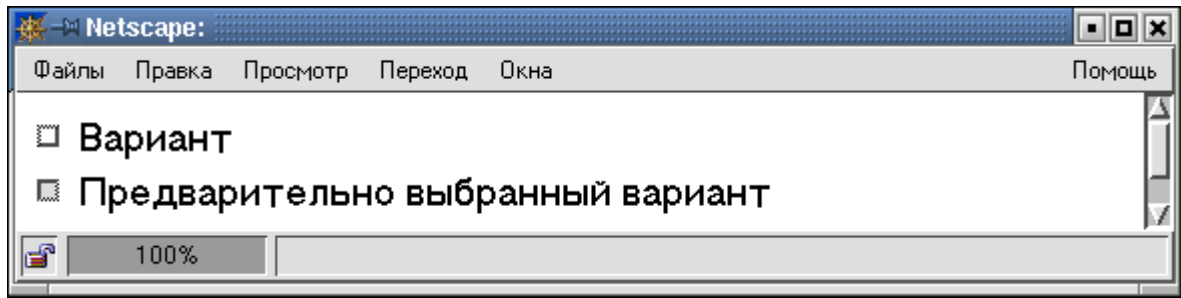
```
<HTML>
<BODY>
<FORM>
Введите секретное слово:
<INPUT TYPE="password" NAME="secret_word"
SIZE="30" MAXLENGTH="30">
</FORM>
</BODY>
</HTML>
```



Пример

Использование значения CHECKBOX.

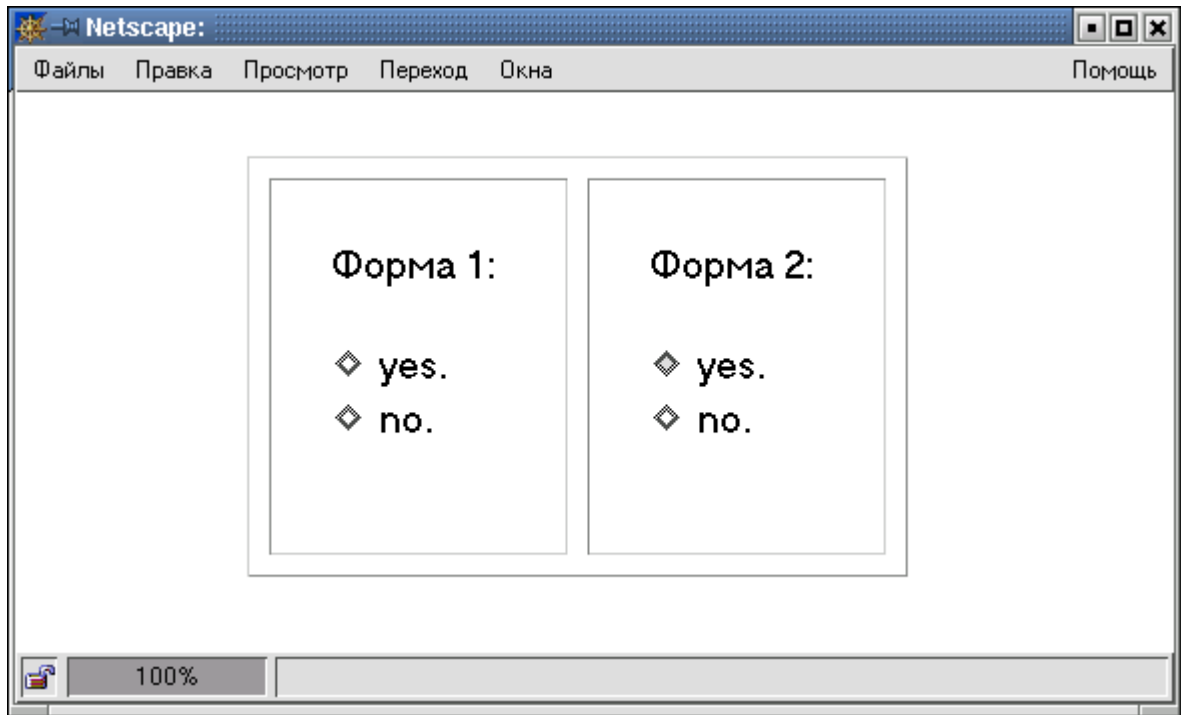
```
<HTML>
<BODY>
<FORM>
<INPUT TYPE="checkbox" NAME="checkbox1"
VALUE="checkbox_value1"> Вариант
<BR>
<INPUT TYPE="checkbox" NAME="checkbox2"
VALUE="checkbox_value2" CHECKED>
Предварительно выбранный вариант
</FORM>
</BODY>
</HTML>
```



Пример

В этом примере две формы расположены в соседних ячейках таблицы.

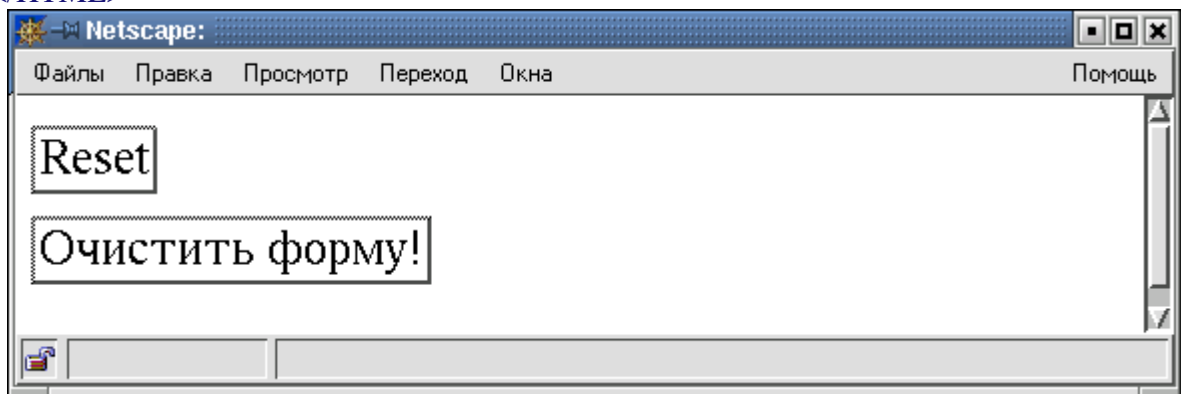
```
<HTML>
<BODY>
<TABLE ALIGN=center BORDER CELLSPACING=10>
<TR>
<TD>
Форма 1:
<FORM>
<INPUT TYPE="radio" NAME="choice"
VALUE="choice1"> yes.
<INPUT TYPE="radio" NAME="choice"
VALUE="choice2"> no.
</FORM>
</TD>
<TD>
Форма 2:
<FORM>
<INPUT TYPE="radio" NAME="choice"
VALUE="choice1" CHECKED> yes.
<INPUT TYPE="radio" NAME="choice"
VALUE="choice2"> no.
</FORM>
</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```



Пример

В следующей форме используется значение `RESET`.

```
<HTML>  
<BODY>  
<FORM>  
<INPUT TYPE="reset">  
<BR>  
<INPUT TYPE="reset" VALUE="Очистить форму!">  
</FORM>  
</BODY>  
</HTML>
```

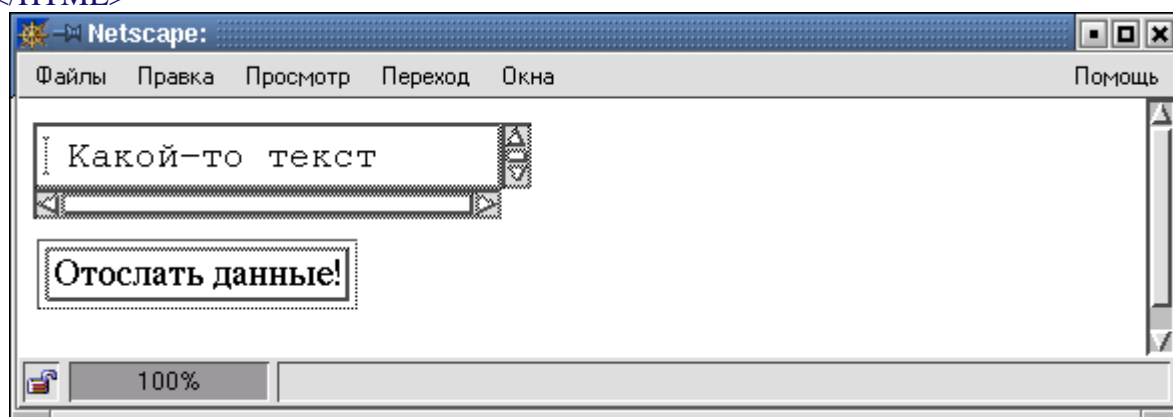


Пример

Использование значения `SUBMIT`.

```
<HTML>  
<BODY>  
<FORM>  
<TEXTAREA> Какой-то текст </TEXTAREA>  
<BR>
```

```
<INPUT TYPE="submit" VALUE="Отослать данные!">
</FORM>
</BODY>
</HTML>
```

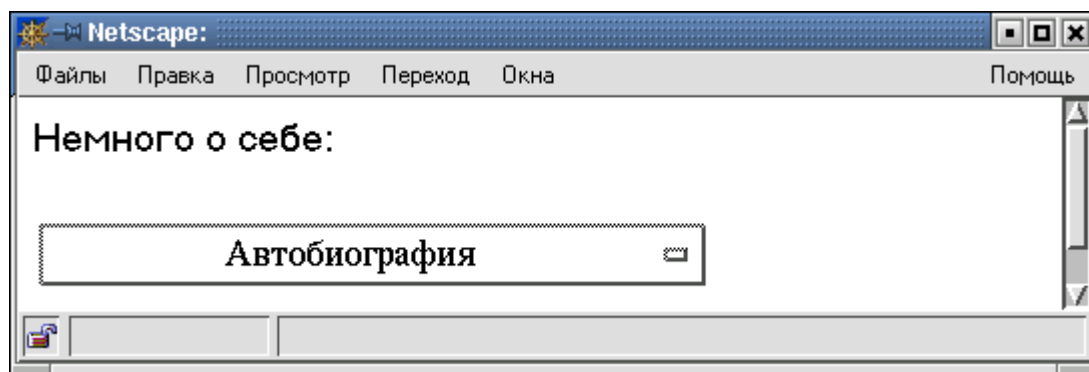


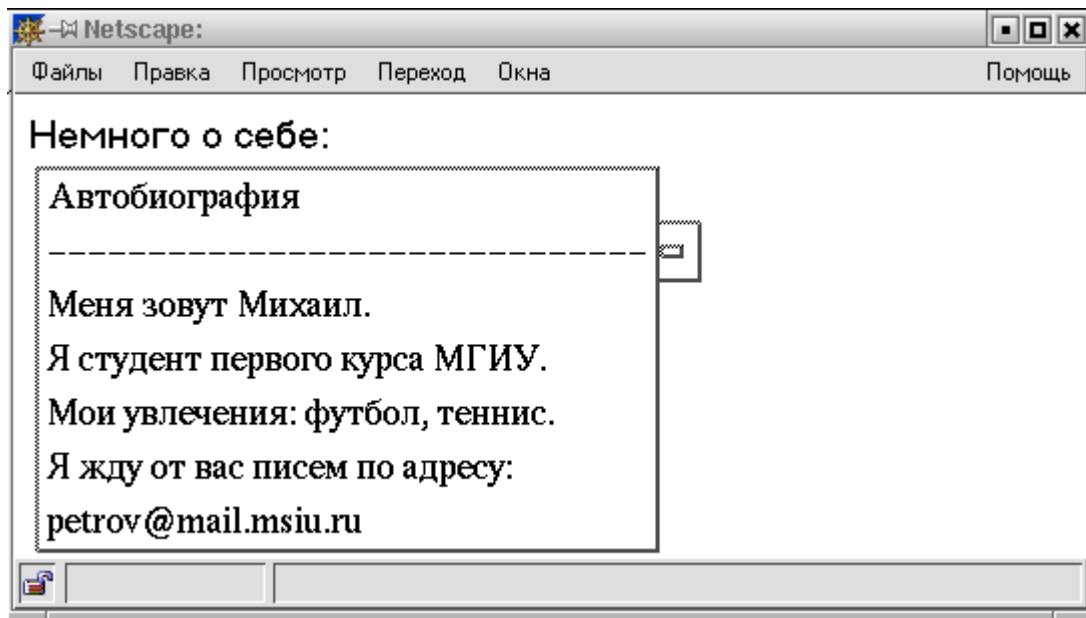
7 Нестандартное использование элементов форм

Многие теги HTML используются в целях, для которых они изначально не предназначались, например, таблицы -- для разметки страниц. Поэтому смело экспериментируйте и с элементами форм.

Наиболее гибкими являются меню выбора. Всплывающее меню может выполнять функцию информационной полосы, включенной в текстовый поток. Например, можно использовать меню выбора для размещения небольших рассказов (помещая фрагменты текста в элементы `OPTION`). Применение этих возможностей заставляет пользователя активнее взаимодействовать с содержимым вашей страницы.

Пример



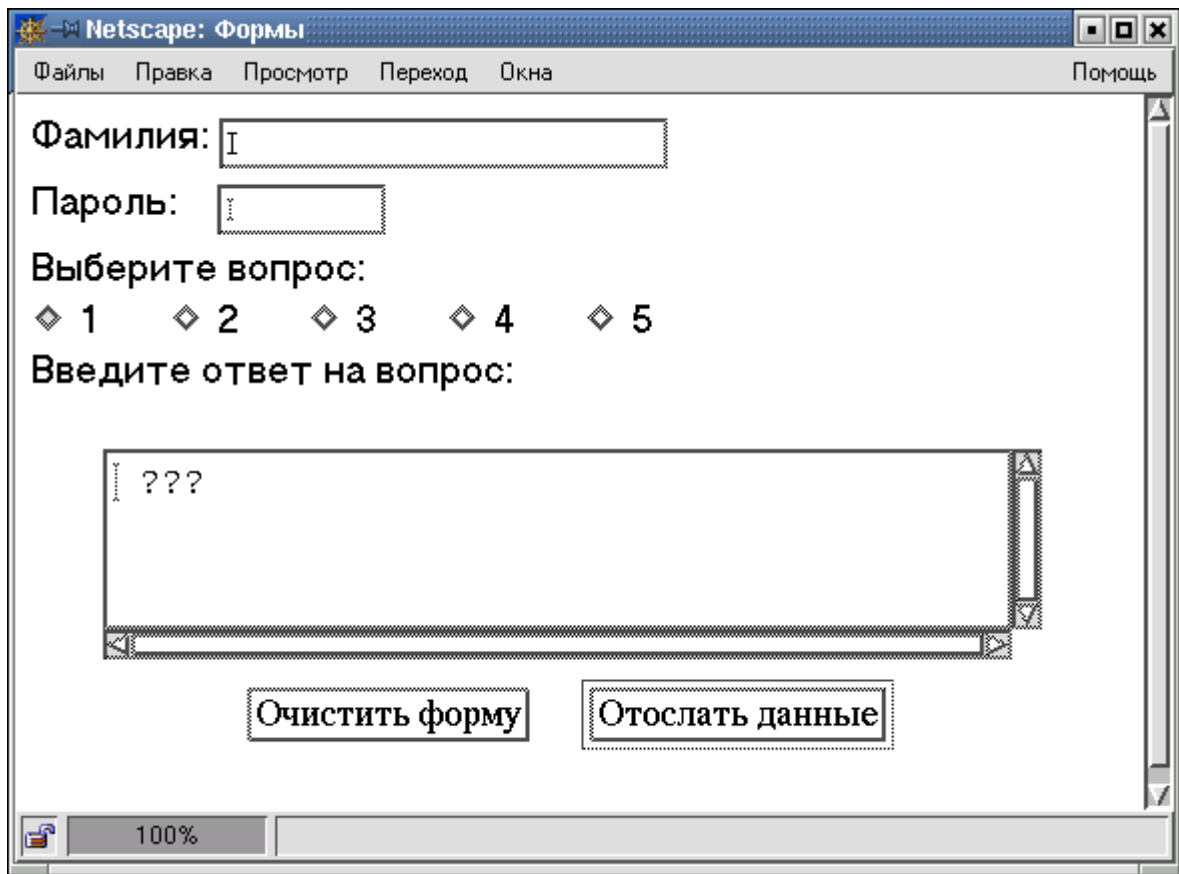


Ниже приведен HTML-документ, в котором раскрывающийся список используется для размещения небольшого рассказа об авторе.

```
<HTML>
<BODY>
Немного о себе:
<FORM>
<SELECT NAME="about">
<OPTION SELECTED VALUE="line0"> Автобиография
<OPTION VALUE="line1"> -----
<OPTION VALUE="line2"> Меня зовут Михаил.
<OPTION VALUE="line3"> Я студент первого курса МГИУ.
<OPTION VALUE="line4"> Мои увлечения: футбол, теннис.
<OPTION VALUE="line5"> Я жду от вас писем по адресу:
<OPTION VALUE="line6"> petrov@mail.msiu.ru
</SELECT>
</FORM>
</BODY>
</HTML>
```

Задание

Создайте файл `form.html` и разместите в нем следующую форму.



Требования к оформлению:

- 1) поле "Пароль" должно иметь размер 10 символов и не отображать введенные данные;
- 2) среди всех номеров вопроса только один может быть выбран, по умолчанию выбранным должен быть вопрос под номером один;
- 3) поле для ответа на вопрос должно содержать четыре строки по сорок символов, первоначальное значение - "???";
- 4) кнопка "Очистить форму" должна восстанавливать первоначальный вид формы.

Практическое занятие №13 Фреймы

Цель занятия: формировать навык по разработке сайта с использованием фреймов.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Один из способов выдать сразу несколько файлов HTML на экран пользователя -- это открыть несколько окон браузера. Другой путь состоит в том, чтобы разбить окно на несколько разделов. Эти разделы называются *фреймами* или кадрами. В каждом фрейме показывается свой HTML-документ. Каждый фрейм может иметь свои полосы прокрутки, ссылки, графические изображения и т. д. Фреймы могут функционировать независимо или влиять друг на друга, используя ссылки, указывающие на другие фреймы.

1 Контейнер <FRAMESET>

Web-страница, которая разделена на фреймы, называется документом группы фреймов. Документы группы фреймов содержат стандартный заголовок, задаваемый тегом **HEAD**, но в отличие от стандартных HTML-документов, они не содержат тега **BODY**. Вместо него используется контейнер (т. е. парный тег) **<FRAMESET>**, который применяется для определения строк и столбцов отдельных фреймов, каждый из которых обозначается тегом **<FRAME>**.

Если включить контейнер **BODY** в документ, где используется контейнер **FRAMESET**, то кадры будут проигнорированы программой просмотра, и информация, содержащаяся в документах, задаваемых тегами **<FRAME>**, не будет выведена. Будет показана только информация, содержащаяся в контейнере **BODY**.

Внутри контейнера **<FRAMESET> ... </FRAMESET>** могут располагаться только теги **<FRAME>** или другие контейнеры **FRAMESET**.

2 Определение параметров кадров

Тег **<FRAMESET>** имеет два главных атрибута: **ROWS** и **COLS**, задающих разбиение на строки и столбцы соответственно. Ниже приведен вид контейнера **FRAMESET**:

```
<FRAMESET ROWS="список_значений" COLS="список_значений">  
...  
</FRAMESET>
```

Можно определить любое число рядов и столбцов; необходимым условием является указание хотя бы одного из атрибутов **ROWS** или **COLS**.

Кадр не может быть единственным: если вы определили единственный ряд и единственный столбец, то программа просмотра проигнорирует контейнер **FRAMESET**, и экран останется пустым. Если определены по крайней мере два ряда или два столбца, другой атрибут может быть опущен (ему по умолчанию будет присвоено значение, равное 100%).

Значение атрибута **ROWS** или **COLS** представляет собой строку, содержащую список значений в пикселах, процентах или относительных единицах, разделенных запятыми. Количество рядов или столбцов кадров определяется числом этих значений.

Пример

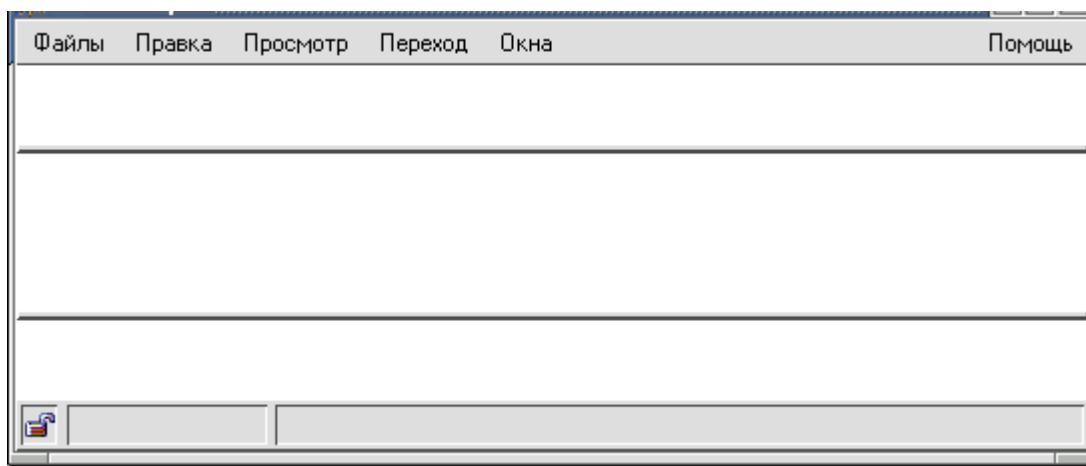
Для задания сетки кадров из трех рядов, высота которых 100, 240 и 140 пикселей соответственно, используйте тег

```
<FRAMESET ROWS="100,240,140">
```

Задание высоты ряда в пикселах, однако, является плохим стилем, так как при этом не учитывается тот факт, что окна браузеров могут иметь самую разную величину. В абсолютных единицах стоит указывать размеры кадра лишь для размещения небольших изображений, в остальных же случаях лучше пользоваться относительными величинами.

Пример

Тег `<FRAMESET ROWS="25%, 50%, 25%">` задаст три кадра, размером по 25%, 50% и 25% от высоты окна браузера.

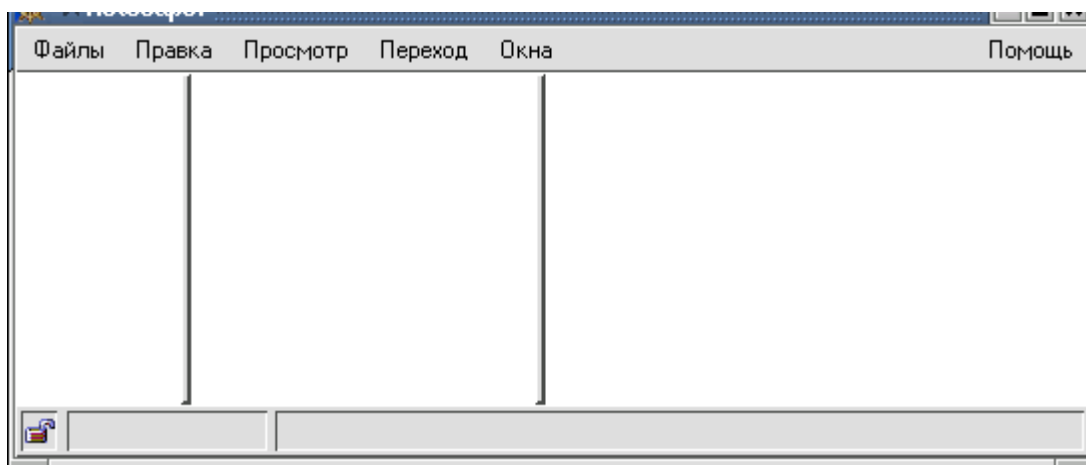


Относительно точности указания размеров фреймов в процентах можно не беспокоиться: если сумма значений не равна 100%, то масштаб кадров будет пропорционально изменен.

Задание параметров кадров в относительных единицах выглядит примерно так:

```
<FRAMESET COLS="*,2*,3*">
```

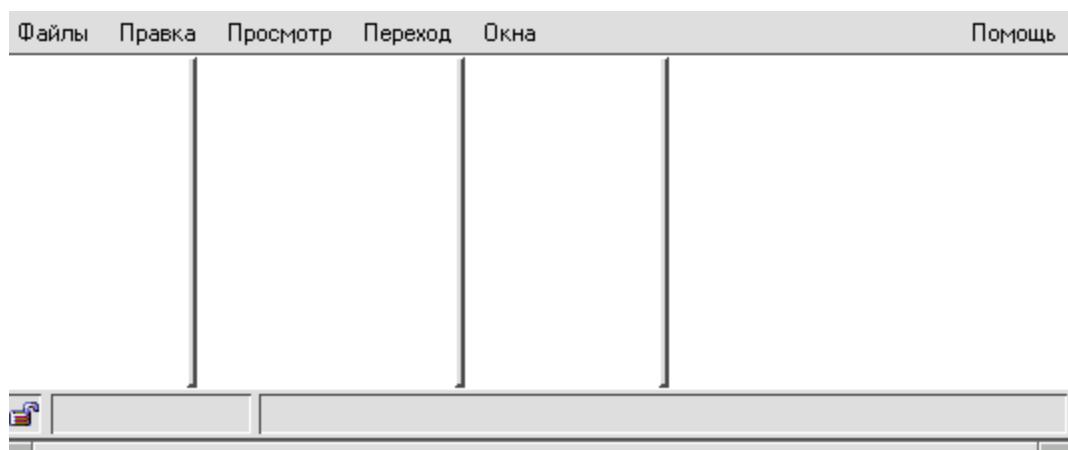
Символ * обозначает пропорциональное деление окна программы просмотра. В данном примере окно будет разделено на три вертикальных кадра, первый из которых будет иметь ширину в 1/6, второй -- в 2/6 (или 1/3) и третий -- в 3/6 (или 1/2) от ширины окна браузера. Единица при указании относительных значений может быть опущена.



Указание значений атрибутов **ROWS** и **COLS** может быть и смешанным, включающим любое сочетание абсолютных размеров, процентных отношений и относительных значений, например,

```
<FRAMESET COLS="100,25%,*,2*">
```

Здесь первому кадру присвоено абсолютное значение в 100 пикселей по ширине, второму -- 25% от ширины окна. Оставшееся пространство делится между третьим и четвертым кадрами в пропорции 1 к 2.



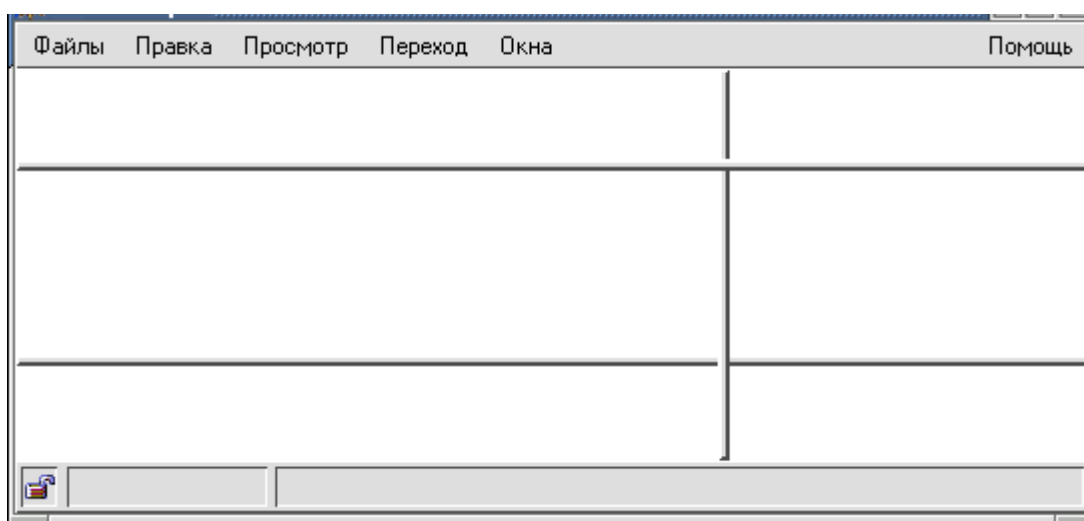
Приоритеты в указаниях значений атрибутов таковы: в первую очередь (слева направо) отводится место для кадра с абсолютным значением, затем -- для кадра со значением в процентах, и в последнюю очередь -- для кадров с относительными величинами.

Если вы пользуетесь абсолютными величинами в атрибутах **ROWS** и **COLS**, не делайте такие кадры большими -- ведь они должны поместиться в окно браузера любого размера. Совместно с такими кадрами для лучшей балансировки рекомендуется использовать хотя бы один кадр, определенный в процентах или в относительных величинах.

При определении обоих атрибутов получается сетка кадров, например, тег

```
<FRAMESET ROWS="*,2*,*" COLS="2*,*">
```

задает сетку из трех рядов и двух столбцов. В данном примере первый и последний ряды занимают по 1/4, второй ряд -- половину от высоты окна. Первый столбец занимает 2/3, а второй -- 1/3 ширины окна браузера.



3 Тег <FRAME>

Тег <FRAME> определяет отдельный кадр. Он должен располагаться внутри контейнера FRAMESET.

Пример

```
<FRAMESET ROWS="*,2*">
<FRAME>
<FRAME>
</FRAMESET>
```

Заметьте, что этот тег не является контейнером и, в отличие от тега <FRAMESET>, не имеет закрывающего тега.

Число тегов <FRAME> **обязательно** должно быть равно числу кадров, определенных в теге <FRAMESET>. В нашем примере определено два кадра, поэтому контейнер содержит соответствующее количество тегов <FRAME>. Пока кадры ничем не заполнены.

В HTML тег <FRAME> располагает шестью атрибутами: SRC, NAME, MARGINWIDTH, MARGINHEIGHT, SCROLLING и NORESIZE. Вот синтаксис использования этих атрибутов:

```
<FRAME SRC="URL" NAME="имя_окна"
SCROLLING=yes|no|auto MARGINWIDTH="значение"
MARGINHEIGHT="значение" NORESIZE>
```

Использовать все атрибуты необязательно. Чаще всего вы будете пользоваться только одним атрибутом -- SRC. Строка

```
<FRAME SRC="URL">
```

определяет URL-адрес содержимого кадра. Это обычно файл HTML-документа, расположенный в том же каталоге, что и документ, содержащий контейнер FRAMESET, например,

```
<FRAME SRC="first.html">
```

Этот документ должен быть полноценным HTML-документом, т. е. содержать все обязательные части (контейнеры HTML, HEAD, BODY и т. д.).

В случае, если программа просмотра не сможет найти указанный файл, кадр не будет построен, и браузер выведет сообщение об ошибке. Если же в теге вовсе не указан атрибут SRC, кадр будет создан и оставлен пустым.

Текст, заголовки, графика и другие элементы не могут напрямую включаться в документ с кадрами. Все они должны вводиться только с помощью указания URL-адреса элемента. Если контейнер FRAMESET содержит "инородное тело", оно будет выведено, а кадры полностью проигнорированы.

Атрибут MARGINWIDTH=n задает размещение слева и справа от содержимого кадра областей свободного пространства высотой по n пикселей, а MARGINHEIGHT=n, соответственно, сверху и снизу.

Значения этих атрибутов всегда должны указываться в абсолютных значениях (пикселах). Так, тег

```
<FRAME MARGINHEIGHT="5" MARGINWIDTH="7">
```

создаст внутреннюю рамку на верхней и нижней границах кадра шириной в 5 пикселей, а на левой и правой границе -- шириной в 7 пикселей. Внутри этой рамки данные выводиться не будут. Эти атрибуты не имеют ничего общего с рамкой кадра, определяемой браузером, либо задаваемой при помощи атрибута **BORDER**.

К построенным вами кадрам автоматически добавляются полосы прокрутки, если содержание кадра больше его размера. Иногда это может нарушить эстетику страницы, поэтому в HTML предусмотрен атрибут **SCROLLING** тега **<FRAME>**, имеющий следующий формат:

```
<FRAME SCROLLING="yes|no|auto">
```

Этот атрибут может принимать одно из трех значений: **yes**, **no** и **auto**. Последнее значение подразумевается по умолчанию, т. е. когда атрибут не определен. Если указано значение **yes**, полоса прокрутки появится в любом случае, значение **no** запрещает появление полосы. Определение атрибута **SCROLLING**, например, может быть следующим:

```
<FRAME SCROLLING=yes>
```

По умолчанию размеры кадров могут легко изменяться читателями, однако понятно, что это может сильно нарушить авторский замысел. Поэтому вы, скорее всего, захотите использовать атрибут **NORESIZE** тега **<FRAME>**, запрещающий возможность "перекраивания" вашей страницы:

```
<FRAME NORESIZE>
```

Этот атрибут не имеет значений. Указав его в одном кадре, вы тем самым запретите изменять размеры и всех смежных кадров. Как правило, этого бывает достаточно, чтобы "закрепить" рамки всех кадров страницы на месте.

Когда вы захватываете рамку кадра мышью, то указатель становится двунаправленной стрелкой, если перемещение рамки возможно. В противном случае, т. е. когда использован атрибут **NORESIZE**, двунаправленная стрелка не появляется.

Для определения рамки кадра в HTML существуют три атрибута: **BORDER**, **FRAMEBORDER** и **BORDERCOLOR**. Первый из этих атрибутов используется только с тегом **<FRAMESET>** и устанавливает ширину всех рамок для всех кадров контейнера **FRAMESET**. Эта величина указывается в пикселах, например,

```
<FRAMESET BORDER="10">
```

Если этот атрибут нулевой, то все кадры контейнера будут без рамок. По умолчанию атрибут **BORDER** имеет значение 5.

Атрибут **FRAMEBORDER** используется с тегами **<FRAMESET>** и **<FRAME>** и может принимать два значения: **yes** или **no**. В случае **yes** рамка имеет трехмерную форму. Если **FRAMEBORDER="no"**, рамка невидима, т. е. она имеет цвет фона окна браузера, устанавливаемого по умолчанию.

По умолчанию атрибут **FRAMEBORDER** имеет значение **yes**, т. е. предусматривает наличие трехмерной рамки. Рамка кадра будет невидимой, если значение **FRAMEBORDER="no"** установлено для всех кадров, смежных с ним.

Атрибут **BORDERCOLOR** может использоваться с тегами **<FRAMESET>** и **<FRAME>**. Ему может быть присвоено стандартное имя цвета или RGB-значение.

Пример

```
<FRAMESET BORDERCOLOR="red" ROWS="*,*">  
<FRAME SRC="first.html" BORDERCOLOR="#FF00FF">  
<FRAME SRC="first.html">  
</FRAMESET>
```

Здесь атрибут **BORDERCOLOR** тега **<FRAMESET>** устанавливает красный цвет рамок ("red"), однако такой же атрибут тега **<FRAME>** отменяет это значение и определяет цвет рамки первого кадра как фиолетовый. В результате второй кадр, в котором цветовой атрибут не определен, будет иметь часть рамки фиолетовой (на стороне, смежной с первым кадром), а остальную часть рамки -- красной.

Если же в двух смежных кадрах определены свои собственные атрибуты **BORDERCOLOR**, то ни один из них не будет иметь силы. Цвет их рамок будет определяться соответствующим атрибутом контейнера **FRAMESET**.

4 Организация ссылок

Теперь, когда мы разобрались с методами создания кадров, познакомимся с их главным предназначением -- управлением навигацией по сайту.

Для определения имени кадра служит атрибут **NAME**. Например, строка **<FRAME NAME="frame1">** создает кадр с именем "frame1", на который можно сделать гипертекстовую ссылку следующим образом:

```
<A HREF="putfirst.html" TARGET="frame1">Нажмите сюда,</A>  
чтобы перейти на первую страницу
```

Атрибут **TARGET** гипертекстовой ссылки содержит имя кадра. При активизации этой ссылки содержимое кадра **frame1**, т. е. файл **first.html**, размещенный в нем при создании, будет заменен файлом **putfirst.html**.

Заметьте, что если атрибут **TARGET** отсутствует, файл **putfirst.html** будет выведен в том же окне или кадре, где находится указатель ссылки. Атрибут **TARGET** как раз и предназначен для указания "цели" -- кадра, в котором должен быть размещен файл, определенный атрибутом **HREF**. Этот принцип замены файлов в одном кадре при управлении этим процессом из другого кадра и лежит в основе навигации по сайту.

Имя	Назначение
_blank	Загружает указанный файл в новое окно без названия
_self	Загружает указанный файл в кадр, откуда делается вызов

<code>_parent</code>	Загружает указанный файл в старший (родительский) кадр сетки кадров; если такой кадр не определен, результат аналогичен действию <code>_self</code>
<code>_top</code>	Загружает указанный файл в полное окно, разрушая всю структуру кадров

Кадру обязательно нужно присвоить имя, иначе на него нельзя будет ссылаться. Поэтому всем кадрам, содержание которых планируется менять, должны быть даны правильные имена. Имена кадров должны начинаться с алфавитно-цифрового символа. Ваши имена не должны начинаться с символа подчеркивания, так как он является первым символом зарезервированных имен кадров, перечисленных в таблице.

Пример

Создайте файл `frames.html` в директории `public_html` и запишите в нем следующие теги:

```
<HTML>
<FRAMESET ROWS="*,*">
<FRAMESET COLS="*,*">
  <FRAME SRC="frame1.html" NAME="fr1">
  <FRAME SRC="frame2.html" NAME="fr2">
</FRAMESET>
  <FRAME SRC="frame3.html" NAME="fr3">
</FRAMESET>
</HTML>
```

После этого, создайте файлы `frame1.html`, `frame2.html`, `frame3.html` и заполните их следующим образом:

frame1.html:

```
<HTML><BODY BGCOLOR=white>
<H1>Frame1</H1>
<A HREF=frame2.html TARGER="fr3">Ссылка на 2 кадр</A>
</BODY></HTML>
```

frame2.html:

```
<HTML><BODY BGCOLOR=red TEXT=yellow>
<H1>Frame2</H1>
<A HREF=frame3.html TARGER=_top>Frame3 во все окно</A>
</BODY></HTML>
```

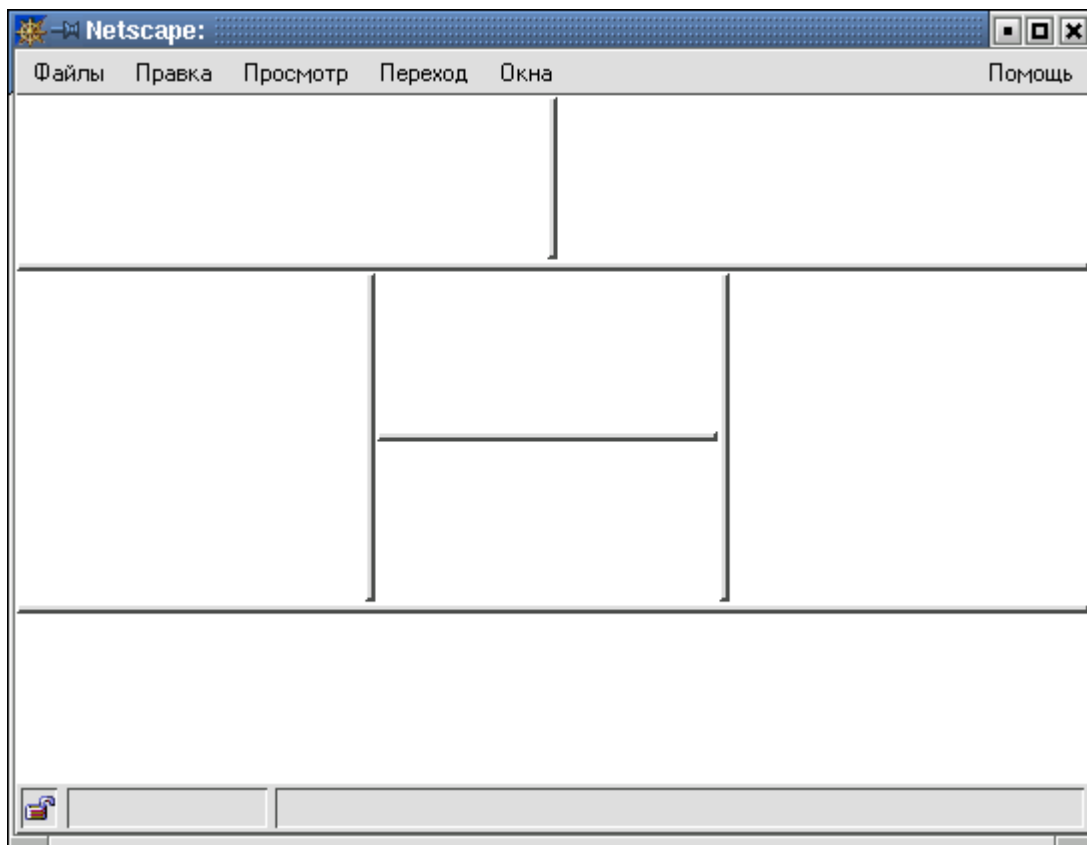
frame3.html:

```
<HTML><BODY BGCOLOR=purple>
<H1>Frame3</H1>
<A HREF=frame1.html TARGER=_self>Frame1 в это окно</A>
</BODY></HTML>
```

Теперь при помощи браузера загрузите файл `frames.html` и посмотрите на результат.

Задания

1. Создайте файл `frame1.html` и с помощью контейнеров `FRAMESET` разбейте окно в соответствии с рисунком.



2. Создайте файл `freym.html`. Разбейте страницу на четыре фрейма (по два в строке и столбце), в которые загрузите ранее созданные файлы: `list.html`, `table.html`, `image.html` и `first.html`. Затем:
 - а) установите в фрейме, содержащим файл `list.html` обязательное наличие полосы прокрутки;
 - б) исключите возможность изменения размера (`NORESIZE`) фрейма, содержащего файл `table.html`;
 - в) измените файл `first.html` таким образом, чтобы при выборе текстовой ссылки на файл `image.html` он загружался в тот же фрейм, где находится сам файл `first.html`;
 - г) в файле `image.html` преобразуйте ссылку в виде картинки на файл `first.html` так, чтобы при нажатии на нее файл `first.html` открывался в новом окне браузера.

Тематика практических и самостоятельных работ по разделу 3 CSS

Тема работы	Вид работы	Содержание занятия	Количество часов
Оформление текста с помощью CSS	Практическое занятие	Занятие № 14 Создание сайта с применением оформления текста с помощью CSS	2
Цвет и фоновое изображение CSS	Практическое занятие	Занятие № 15 Работа с цветом фона и фоновыми изображениями	2
	Самостоятельная работа	Оформление индивидуального сайта	3
Модель компоновки	Практическое занятие	Занятие № 16 Применение команд CSS для компоновки элементов сайта	2
	Самостоятельная работа	Применение команд CSS для компоновки элементов индивидуального сайта	2
Оформление списков и ссылок CSS	Практическое занятие	Занятие № 17 Оформление ссылок и списков сайта	2
	Самостоятельная работа	Добавление списков и ссылок CSS на индивидуальный сайт	3
Оформление таблиц с помощью CSS	Практическое занятие	Занятие № 18 Оформление таблиц сайта с помощью CSS	2
	Самостоятельная работа	Добавление таблиц на индивидуальный сайт	3
Позиционирование в CSS	Практическое занятие	Занятие № 19 Управление размещением структурных блоков и их содержимого. Проверка HTML-документов и CSS-кодов	2
	Самостоятельная работа	Применение команд CSS для позиционирования элементов индивидуального сайта	3

Практическое занятие №14 Оформление текста с помощью CSS

Цель занятия: формировать навык создания сайта с применением оформления текста с помощью CSS.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Задание свойств шрифтов

Существуют тысячи шрифтов, которые предназначены для оформления текстов. Однако, число шрифтов, применяемых для набора текста на сайтах, существенно ниже. Конечно, можно задать, например, для заголовка, вычурный *шрифт*, установленный на компьютере разработчика. Но если такого шрифта на компьютере пользователя нет, то текст будет отображаться шрифтом, установленным в браузере по умолчанию. Получается, что труд дизайнеров и разработчиков пропал даром.

Одной из возможностей обойти эту проблему является новая концепция стандарта *CSS2*. В основе новой концепции лежит понятие загружаемых шрифтов, т.е. шрифтов, отсутствующих на компьютере пользователя, но доступных для загрузки из сети *Интернет*. В *дополнение* к этому *CSS2* предусматривает наличие *базы данных* о шрифтах, содержащей их разнообразные характеристики и позволяющей по мере необходимости синтезировать недостающие шрифты на основе шрифтов, доступных обозревателю. Однако, несмотря на то, что *CSS* поддерживает эту возможность, в реальности она используется очень редко, т.к. далеко не все браузеры поддерживают данную технологию, а пользователи не любят загружать лишнюю информацию.

Поэтому самым распространенным способом гарантировать правильное *отображение* шрифтов в браузере пользователя является использование стандартных шрифтов, встроенных в *браузер* и операционную систему.

Семейство шрифтов: свойство *font-family*

Свойство *font-family* используется для задания списка имен семейств шрифтов для отображения содержимого элемента. Список шрифтов может включать одно или несколько названий, разделенных запятыми. Если в имени шрифта содержатся пробелы, например, Times New Roman, оно должно заключаться в двойные или одинарные кавычки. Гарнитурные названия должны указываться в порядке возрастающей вероятности доступности или предпочтения. В *качестве* защиты от отказа значение свойства *font-family* всегда должно заканчиваться ключевым словом, ссылающимся на родовое имя шрифта. Таким образом, последовательность шрифтов лучше начинать с экзотических типов и заканчивать обобщенным именем, которое задает вид начертания.

Например, следующее ниже свойство следует понимать как указание браузеру пользователя использовать шрифт Verdana; если его нет, то использовать шрифт Arial; если его нет, то использовать родовой шрифт *sans-serif*:

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

Такой список необходим, поскольку разработчику заранее не известно, какие именно шрифты установлены на компьютерах пользователей.

Имя *семейства шрифтов* может быть задано как название *семейства шрифтов* (например, Times New Roman, Arial и т.д.) или как родовое имя. Родовые имена

шрифтов были разработаны на тот случай, если на компьютере пользователя не установлен ни один из шрифтов, заданных разработчиком. В этом случае браузер использует родовой шрифт, начертание которого напоминает шрифт, который планировал использовать разработчик. Спецификацией определено пять родовых имен, изображения которых представлены на рисунке 14.1.

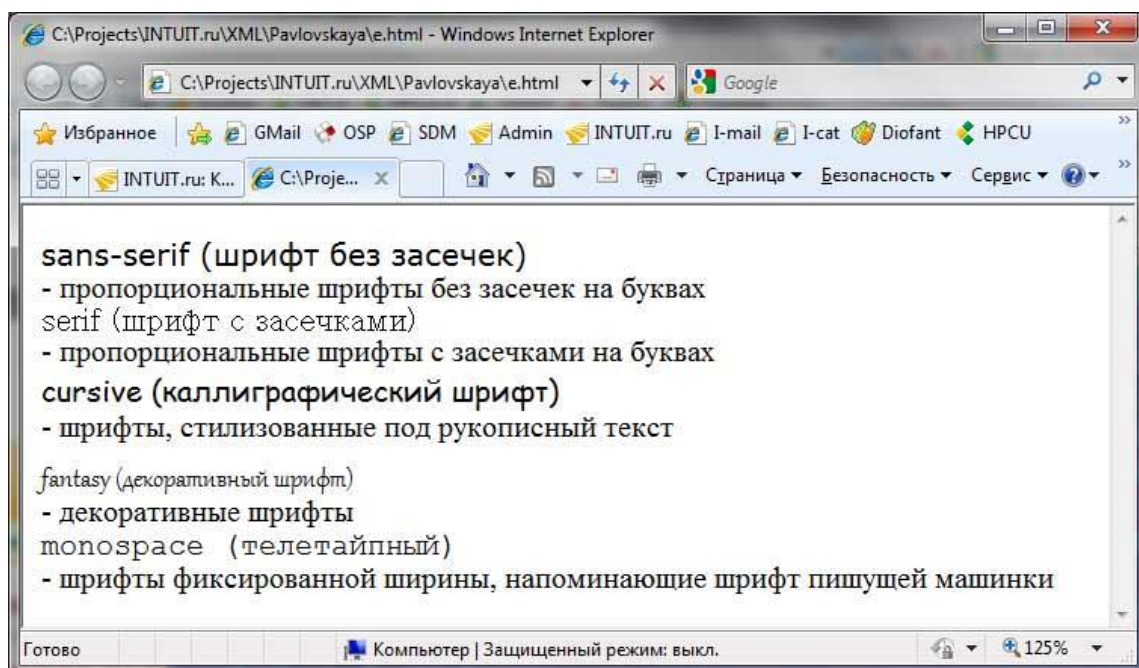


Рис. 14.1. Пример различных семейств шрифтов

Т.к. список шрифтов на компьютерах пользователей может сильно различаться в зависимости от операционной системы и собственных предпочтений, необходимо пользоваться наиболее распространенными шрифтами, к которым относятся Arial, Comic Sans MS, Courier, Courier New, Lucida Console, Tahoma, Times, Times New Roman, Trebuchet MS, Verdana. Однако следует помнить, что шрифты с одинаковыми именами в разных браузерах и системах могут незначительно отличаться друг от друга по форме или по размеру.

Размер шрифтов: свойство font-size

Размер шрифта может быть установлен несколькими способами. Набор констант **xx-small**, **x-small**, **small**, **medium**, **large**, **x-large**, **xx-large** задает размер, который называется абсолютным. По правде говоря, он не совсем абсолютный, поскольку зависит от настроек браузера и операционной системы. На рисунке 14.2 представлены варианты размеров шрифтов, соответствующих данным константам.

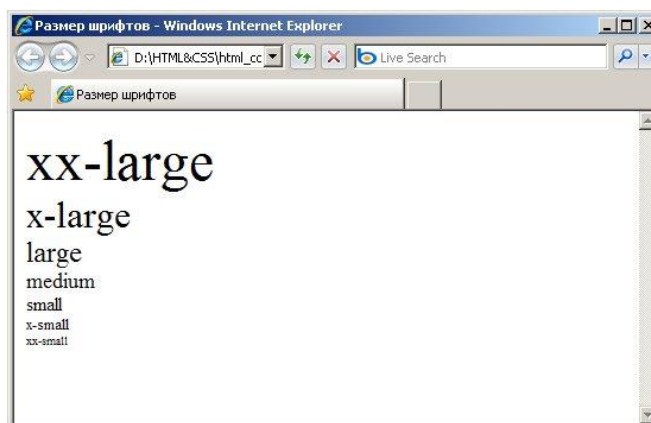


Рис. 14.2. Пример использования различных значений свойства font-size

Другой набор констант `larger`, `smaller` устанавливает *относительные размеры* шрифта. Поскольку размер унаследован от родительского элемента, эти *относительные размеры* применяются к родительскому элементу, чтобы определить размер шрифта текущего элемента.

Также разрешается использовать любые допустимые единицы CSS: `em` (высота шрифта элемента), `ex` (высота символа `x`), пункты (`pt`), пиксели (`px`), проценты (%) и др. При использовании процентной записи за 100% принимается размер шрифта родительского элемента. Если размер шрифта задается в пунктах или пикселях, то изменить эту величину с помощью специальной опции браузера "Размер шрифта" нельзя. Если шрифт установлен слишком мелким, то исправить этот недостаток пользователю простыми средствами не удастся. Поэтому лучше использовать другие единицы размеров шрифта, например, проценты.

Насыщенность шрифтов: свойство font-weight

Насыщенность шрифта (или жирность) управляется с помощью свойства *font-weight*. Значениями этого свойства могут быть ключевые слова **bold**, **bolder**, **lighter** и **normal**, которые устанавливают полужирное, жирное, светлое и нормальное начертание шрифта. Также можно использовать условные единицы от 100 до 900 с шагом 100, причем чем больше значение, тем выше жирность. Установленное по умолчанию нормальное начертание шрифта эквивалентно значению 400, а стандартный полужирный текст - 700. Задание насыщенности шрифта может выглядеть следующим образом:

```
P {font-weight: 900;}
```

Стиль шрифта: свойство font-style

Свойство *font-style* определяет начертание шрифта как обычное, курсивное или наклонное. Данным начертаниям соответствуют значения свойства **normal**, **italic** и **oblique**. Когда для текста установлено курсивное или наклонное начертание, браузер обращается к системе для поиска подходящего шрифта. Если заданный шрифт не найден, браузер использует специальный алгоритм для имитации нужного вида текста. Результат и качество при этом могут получиться неудовлетворительными, особенно при печати документа.

Капитель: свойство font-variant

Капителью называется текст, набранный прописными буквами уменьшенного размера. Для создания такого эффекта используется свойство *font-variant* со значением **small-caps**. Особенность капители заключается в том, что заглавные и строчные буквы при ее использовании сохраняются. Браузер Internet Explorer до шестой версии отображает текст неправильно, заменяя все символы прописными. Остальные браузеры преобразуют символы вполне корректно.

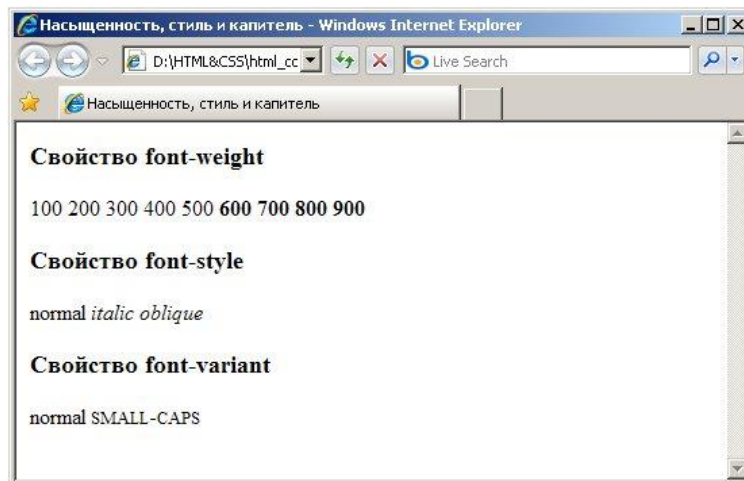


Рис. 14.3. Пример использования различных значений свойств font-weight, font-style и font-variant

Задание свойств текста

Преобразование текста: свойство text-transform

Обычно создатель сайта сам решает, какие буквы будут прописными, а какие строчными, исходя из правил правописания и собственных предпочтений. Тем не менее, процесс изменения регистров символов можно автоматизировать, используя свойство *text-transform*. Данное свойство может принимать четыре значения:

- *none* - текст пишется без изменений;
- *capitalize* - каждое слово будет начинаться с заглавного символа;
- *lowercase* - все символы становятся строчными (нижний регистр);
- *uppercase* - все символы становятся прописными (верхний регистр).

Например, следующее правило указывает, что заголовок **H1** должен выводиться прописными буквами:

```
H1 {text-transform: uppercase;}
```

Автоматическое изменение регистра символов удобно задавать для аббревиатур, названий элементов, заголовков и других *элементов текста*, где требуется писать прописными или строчными символами.

Украшение текста: свойство text-decoration

Свойство *text-decoration* позволяет задать тексту дополнительное оформление. Значениями данного свойства являются константы *none*, *underline*, *overline*, *line-through* и *blink*, позволяющие отобразить обычный текст, провести линию над, под или через текст, а также сделать текст мигающим. Пример использования различных значений данного свойства приведен на [рисунке 14.4](#).

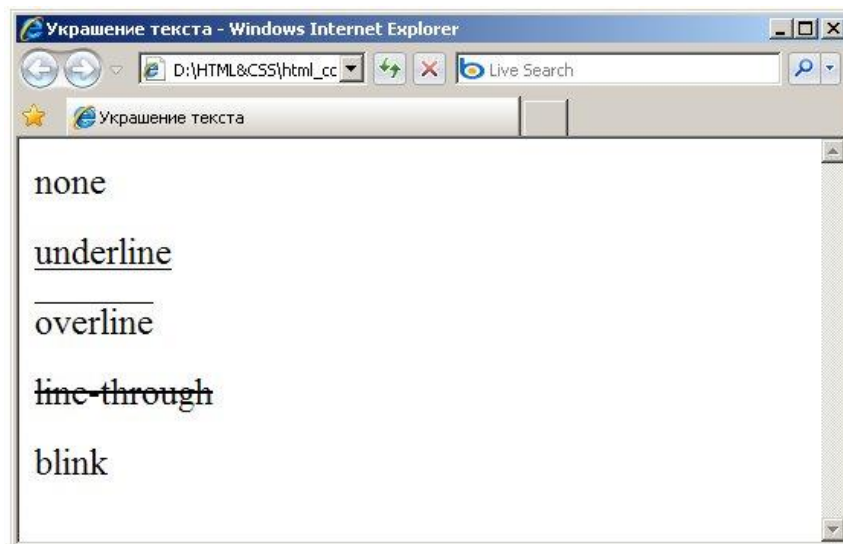


Рис. 14.4. Пример использования различных значений свойства `text-decoration`

Наиболее распространенным применением свойства `text-decoration` является изменение используемого по умолчанию подчеркивания ссылок. Например, следующее правило указывает, что ссылки должны подчеркиваться:

```
A:link {text-decoration: underline;}
```

Интервал между словами: свойство `word-spacing`

Чтобы задать интервал между словами в тексте, используется свойство `word-spacing`. Значения данного свойства можно задать с помощью ключевого слова `normal`, которое используется по умолчанию и задает стандартный интервал для текущего шрифта. Для задания интервала в дополнение к стандартному можно указать значение в любых доступных единицах CSS, причем значение может быть и отрицательным.

Так, следующее правило увеличивает интервал между словами в заголовке `H1` на `1em`:

```
H1 {word-spacing: 1em;}
```

Выравнивание текста: свойство `text-align`

Выравниванием называется размещение левого или правого края блока текста вдоль невидимой вертикальной линии. Для выравнивания текста используется свойство `text-align`. Допустимыми значениями данного свойства являются `left`, `right`, `center` и `justify`, задающие выравнивание по левому краю, по правому краю, по центру и по ширине соответственно.

Следующее правило устанавливает выравнивание по центру всех элементов, содержащихся в элементе `DIV`:

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

Интерлиньяж: свойство `line-height`

Интерлиньяжем называется расстояние между базовыми линиями близких друг к другу строк. При обычных обстоятельствах расстояние между строками зависит от вида и размера шрифта и автоматически определяется браузером. Но это значение может быть изменено с помощью свойства `line-height`. Заданное по умолчанию значение `normal` заставляет браузер вычислять расстояние между строками автоматически. Любое число больше нуля

воспринимается как множитель от размера шрифта текущего текста. В качестве значений данного свойства допустимо также использовать любые единицы длины, принятые в CSS. Разрешается также использовать процентную запись, причем в этом случае за 100% принимается высота шрифта. Отрицательное значение межстрочного расстояния не допускается.

Интервал между буквами: свойство `letter-spacing`

Браузер автоматически подбирает интервалы между символами, исходя из размера и типа шрифта. В некоторых случаях необходимо подкорректировать расстояние между буквами. Для управления межбуквенным интервалом используется свойство `letter-spacing`. В качестве значений данного свойства могут использоваться любые единицы длины, принятые в CSS, однако рекомендуется использовать относительные единицы, основанные на размере шрифта (`em` и `ex`). В отличие от межстрочного интервала, свойство `letter-spacing` допускает использование отрицательного значения, однако в этом случае надо убедиться, что сохраняется читабельность текста.

Следующее правило увеличивает интервал между символами в заголовке `H1` на `0.5em`:

```
H1 {letter-spacing: 0.5em;}
```

Задание

1. К созданному ранее файлу `index.html` учебного сайта применить свойства настройки шрифта и текста.

Практическое занятие №15 Цвет и фоновое изображение CSS

Цель занятия: формировать навык при работе с цветом фона и фоновыми изображениями.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

1 Управление цветом переднего плана: свойство color

Цвет текста задается свойством color. Значения данного свойства можно задавать несколькими способами. Можно задать явное название цвета (например, red, yellow и др.), указать шестнадцатеричное значение или значение RGB. Шестнадцатеричное значение состоит из символа #, за которым следует шесть символов. Первая пара указывает уровень красного цвета, а вторая и третья – уровни зеленого и синего цветов соответственно, например, #FF0000. Можно определить цвет, используя значения уровня красной, зеленой и синей составляющей в десятичном исчислении, например, RGB(49, 151, 116). Также можно задавать цвет в процентном отношении. Например, следующее свойство делает все заголовки документа красными, а для задания свойства используется шестнадцатеричное значение:

```
H1 {color: #FF0000;}
```

Некоторые программы позволяют выбрать оттенок цвета, а затем определить его шестнадцатеричное или RGB-значение, как это показано на рисунке 15.1.

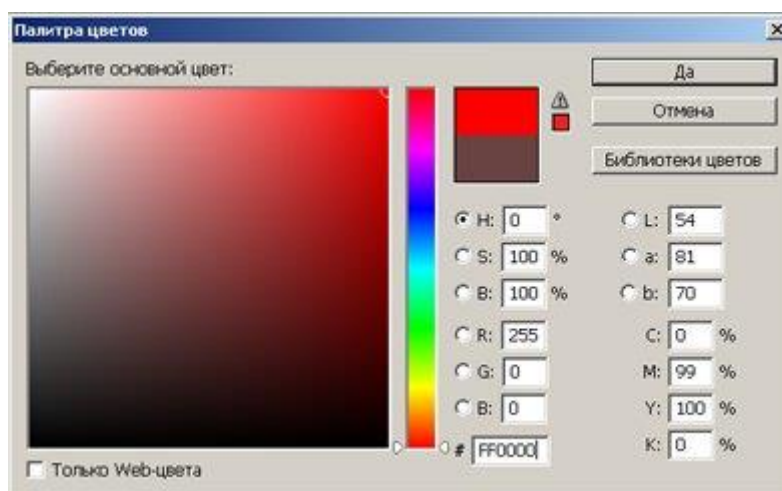


Рис. 15.1. Пример выбора значения цвета в программе Adobe Photoshop

Далее представлены основные свойства CSS, предназначенные для оформления фона элемента.

2 background

Синтаксис

```
background: background-attachment || background-color || background-image || background-position || background-repeat
```

Аргументы

Любые комбинации пяти значений, определяющих стиль фона, в произвольном порядке. Значения разделяются между собой пробелом. Ни один аргумент не является обязательным, поэтому неиспользуемые значения можно опустить.

Значение по умолчанию

Нет.

Наследование

Значения, присвоенные данному параметру, не наследуются.

Пример

```
BODY { background: url(bg.gif ) repeat fixed }  
P { background: #C0C0C0 }
```

Применяется

Ко всем элементам.

3 background-attachment

Описание

Параметр `background-attachment` устанавливает, будет ли прокручиваться фоновое изображение вместе с содержимым элемента. Изображение может быть зафиксировано и оставаться неподвижным, либо перемещаться совместно с документом.

Синтаксис

```
background-attachment: fixed | scroll
```

Аргументы

Значение `fixed` делает фоновое изображение элемента неподвижным, `scroll` — позволяет перемещаться фону вместе с содержимым.

Значение по умолчанию

`scroll`

Наследование

Значения, присвоенные данному параметру, не наследуются.

Пример

```
BODY { background-image: url(images/bg.gif);background-attachment: fixed }
```

Применяется

Ко всем элементам.

Описание

Устанавливает фоновый цвет элемента. Хотя этот параметр не наследует свойства своего родителя, из-за того, что начальное значение цвета фона устанавливается прозрачным, он совпадает с фоном текущего элемента.

4 background-color

Синтаксис

```
background-color: цвет
```

Аргументы

Значение цвета может задаваться по его названию, шестнадцатеричному значению, либо с помощью RGB. Кроме значения цвета, еще один допустимый аргумент — transparent, устанавливающий прозрачный фон.

Значение по умолчанию

transparent

Наследование

Значения, присвоенные данному параметру, не наследуются.

Пример

```
BODY { background-color: #3366CC }  
P { background-color: yellow }  
P { background-color: #98560F }  
P { background-color: RGB(249, 231, 16) }
```

Применяется

Ко всем элементам.



Рис. 15.1. Применение цвета фона для элементов BODY и H1

5 background-image

Описание

Устанавливает фоновое изображение для элемента. Если одновременно для элемента задан цвет фона, он будет показан, пока фоновая картинка не загрузится полностью. То же произойдет, если изображение не доступно или отключен их показ в браузере. В случае наличия в рисунке прозрачных областей, через них будет проглядывать фоновый цвет.

Синтаксис

```
background-image: url(путь к файлу) | none
```

Аргументы

В качестве значения используется путь к графическому файлу, который указывается внутри

конструкции url(). Либо аргумент может принимать значение none, когда фоновое изображение не требуется.

Значение по умолчанию

none

Наследование

Значения, присвоенные данному параметру, не наследуются.

Пример

```
BODY { background-image: url(images/bg.gif) }
```

Применяется

Ко всем элементам.

6 background-position

Описание

Задаёт положение левого верхнего угла фонового изображения, установленного с помощью параметра background-image.

Синтаксис

```
background-position: [проценты | значение] | [left | center | right] || [top | center | bottom]
```

Аргументы

У этого параметра два значения, положение по горизонтали (может быть — left, center, right) и вертикали (может быть — top, center, bottom). Положение можно, также, задавать в процентах, пикселях или других единицах.

Значение по умолчанию

0%

Наследование

Значения, присвоенные данному параметру, не наследуются.

Пример

```
BODY { background-image: url(mybg.gif); background-position: right bottom }
```

Применяется

Ко всем элементам.

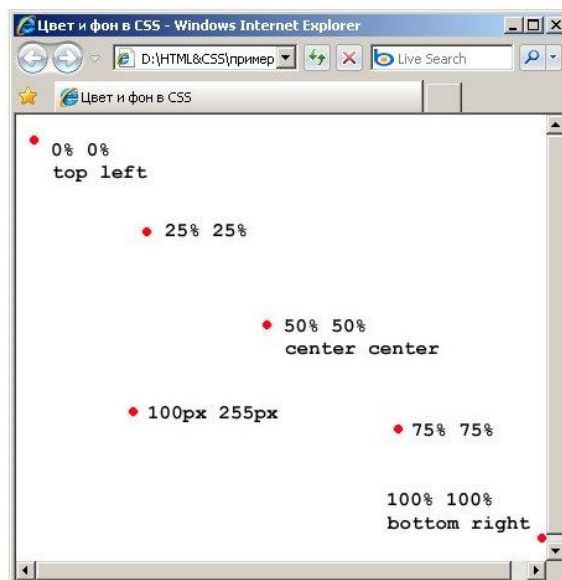


Рис. 15.2. Различные примеры позиций фонового изображения, использующие ключевые слова, проценты и пиксели

7 background-repeat

Описание

Определяет, как будет повторяться фоновое изображение, установленное с помощью параметра `background-image`, и по какой оси. Можно установить повторение рисунка только по горизонтали, по вертикали или в обе стороны.

Синтаксис

`background-repeat: no-repeat | repeat | repeat-x | repeat-y`

Аргументы

Значение `no-repeat` устанавливает одно фоновое изображение в элементе без его повторений, положение которого определяется атрибутом `background-position` (по умолчанию в левом верхнем углу). Другими допустимыми значениями являются `repeat` (повторяемость по вертикали и горизонтали), `repeat-x` (по горизонтали), `repeat-y` (по вертикали).

Значение по умолчанию

`repeat`

Наследование

Значения, присвоенные данному параметру, не наследуются.

Пример

```
BODY { background-image: url(mybg.gif); background-repeat: repeat-y }
```

Применяется

Ко всем элементам.

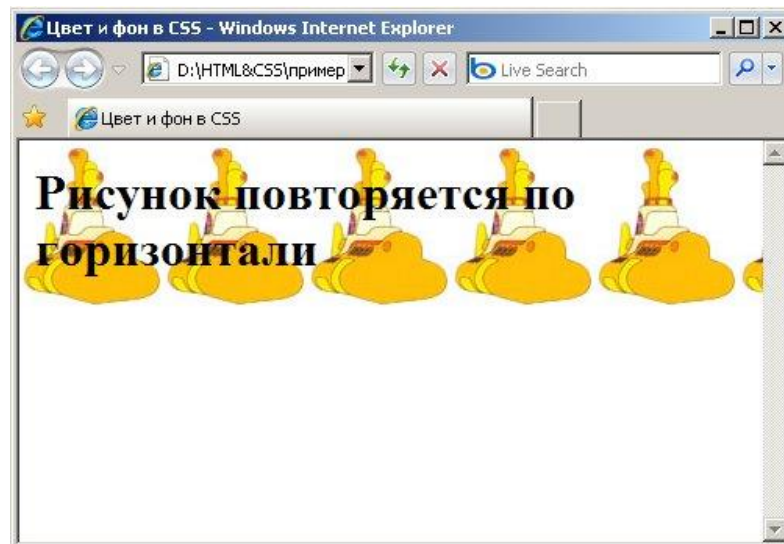


Рис. 15.3. Повторение фонового рисунка по горизонтали

Задание

Оформить созданные на предыдущих практических занятиях страницы, добавив *цвет фона* и фоновое изображение. Для фонового изображения необходимо предусмотреть такие свойства как направление повторения фонового изображения и поведение изображения при прокрутке. Также будет полезно оформить фоном не только саму страницу (т.е. элемент **BODY**), но и некоторые структурные элементы, например, *заголовки* или параграфы.

При оформлении текста необходимо установить такие свойства как семейство шрифтов, размер, насыщенность и стиль, капитель и другие свойства, описанные в "[Оформление текста с помощью CSS](#)". Данные свойства необходимо применять как к блокам текста, так и к группам символов. Слушателю предлагается поработать над цветовым оформлением текста и отдельных его символов. Выбор объектов применения данных свойств оставляется на усмотрение пользователя.

Задание потребует предварительной группировки элементов, созданного HTML-документа, в блоки с использованием элементов **DIV** и **SPAN**.

Практическое занятие №16 Модель компоновки

Цель занятия: формировать навык применения команд CSS для компоновки элементов сайта.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Одним из самых важных инструментов дизайна является точное управление свободным пространством. Любое пустое *пространство* вокруг элемента невольно притягивает к нему взгляд, а для текста еще и обеспечивает его оптимальное восприятие. Пустой промежуток вокруг элемента выделяет его на веб-странице и позволяет отделить один элемент от другого. Однако таблицы стилей браузера, используемые *по умолчанию*, не решают задачу управления свободным пространством с достаточной точностью, поэтому разработчикам часто приходится использовать поля, границы, заполнение и другие свойства компоновки CSS. Все свойства компоновки CSS объединены в модель компоновки CSS, которая также называется боксовой моделью. Боксовая модель имеет детальные опции для определения полей, границ, заполнения и содержимого каждого элемента. Однако прежде чем рассматривать свойства для оформления боксов, необходимо немного поговорить о них самих.

Документ *HTML* состоит из множества перемешанных элементов. Когда такой документ изображается на экране компьютера или печатается на бумаге, эти элементы генерируют прямоугольные боксы. По умолчанию, встроенная *таблица стилей* в браузере заставляет элементы *HTML* блочного уровня (такие, как **P** и **DIV**) генерировать блочные боксы, в то время как строковые элементы (такие, как **STRONG** и **SPAN**) генерируют строковые боксы. Типом генерируемого бокса можно управлять, используя свойство **display**, которое будет рассмотрено ниже.

На рисунке показано, как построена боксовая модель:



Рис. 16.1. Иллюстрация различных частей бокса элемента, помеченных соответствующими свойствами CSS

Поля элемента: свойство **margin**

Для управления значениями полей элементов предназначено свойство **margin**. Это универсальный параметр, в зависимости от числа значений, он устанавливает поля со всех сторон элемента или для каждой его стороны отдельно. Например, указание одного значения задаст равные поля вокруг элемента.

Допустимые значения обычно определяют в единицах измерения px или em (пикселях или em). В таблицах стилей, предназначенных для печати, в качестве единиц измерения можно использовать дюймы (in), сантиметры (cm) или пункты pt (пункты).

Для задания полей с разных сторон элемента предназначены производные от свойства `margin` - `margin-left`, `margin-right`, `margin-top` и `margin-bottom`, задающие значения левого, правого, верхнего и нижнего поля соответственно. Например, ниже представлен пример задания полей документа, т.е. элемента **BODY**. На рисунке 13.2 показано, какие поля необходимо определить и какие значения необходимо им придать.

```
BODY {  
  margin-top: 100px;  
  margin-right: 70px;  
  margin-bottom: 40px;  
  margin-left: 40px;  
}
```

Это же правило можно записать в следующем виде:

```
BODY {margin: 100px 70px 40px 40px;}
```

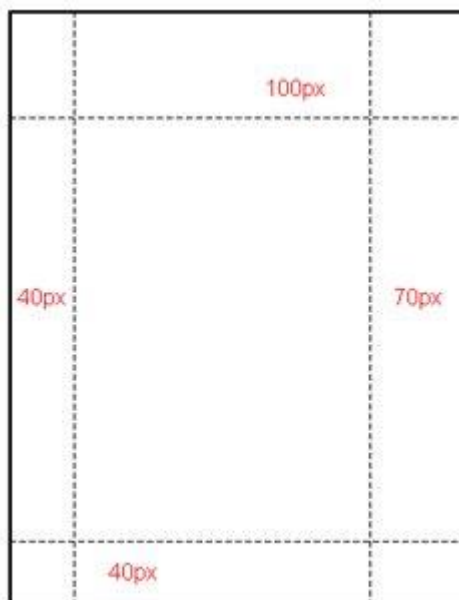


Рис. 16.2. Иллюстрация применения свойства `margin`

Таким же образом можно установить поля почти для любого элемента. Например, можно определить поля для всех параграфов на веб-странице:

```
P {margin: 5px 50px 5px 50px;}
```

Добавление границы: свойства `border`, `border-width`, `border-style` и `border-color`

Границы имеют разнообразное применение, например, как декоративный элемент или для отделения двух объектов. Для задания границ применяется несколько способов, один из которых основан на использовании свойства `border` и его производных. Это свойство позволяет одновременно установить толщину, стиль и цвет границы вокруг элемента. Значения разделяются пробелами и могут идти в любом порядке. Браузер сам определит, какое значение соответствует нужному атрибуту:


```
P {border: 2px solid black;}
```

Данное правило позволяет создать вокруг прямоугольной области сплошную рамку черного цвета толщиной 2 пиксела. Первый аргумент в данном случае определяет толщину, второй - тип линии, а третий - ее цвет.

Когда значение в `border` отсутствует, выводимый элемент будет использовать значения по умолчанию: толщина границы будет определяться браузером, стиль границы будет **solid**, а цвет границы будет совпадать с цветом, используемым для рассматриваемого элемента.

Можно задать толщину, стиль и цвет любой из четырех сторон элемента, используя свойства **border-top**, **border-bottom**, **border-left** и **border-right**. Например, следующий пример создает нижнюю границу для элемента **H1** в виде красной сплошной линии толщиной 1 пиксел:

```
H1 {border-bottom: 1px solid red;}
```

Толщину, стиль и цвет также можно задать отдельно, используя соответствующие свойства.

Толщина границы: свойство **border-width**

Это свойство задает толщину одной или нескольких сторон границы. Сокращенное свойство **border-width** принимает значения в той же нотации, что и сокращенное свойство **margin**, за исключением того, что процентные значения не поддерживаются. Например, свойство **border-width** может быть задано следующим образом:

```
TD {border-width: 1px 0 0 1px;}
```

Стиль границы: свойство **border-style**

Свойство **border-style** задает стиль линии и может принимать одно из восьми значений, представленных на [рисунке 13.3](#).

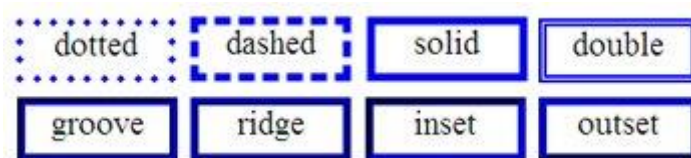


Рис. 16.3. Возможные значения свойства **border-style**

Цвет границы: свойства **border-color**

Для каждой границы можно задать любой цвет с помощью сокращенного свойства **border-color** или его уточнения. Например

```
TD {border-color: #FF0000;}
```

Заполнение элемента: свойство **padding**

Заполнение определяет внутреннее расстояние между границей и содержимым элемента. Для изменения этой характеристики предназначено свойство **padding**. Оно позволяет задать расстояние между границей и содержанием для всех или определенных сторон элемента. Это свойство действует аналогично **margin**, поэтому итоговый результат зависит от числа аргументов. Для указания расстояний от разных сторон элемента можно воспользоваться свойствами **padding-left**, **padding-right**, **padding-top** и **padding-bottom**, которые управляют величиной расстояния слева, справа, сверху и снизу соответственно.

Основное предназначение заполнения - создать пустое пространство вокруг содержимого *блочного элемента*, например, текста, чтобы он не прилегал плотно к границе элемента. Использование заполнения повышает читабельность текста и улучшает внешний вид страницы. В следующем примере показано использование заполнения для оформления текста:

```
DIV.first {padding: 20px;}
```

```
DIV.second {padding: 10px;  
padding-left: 50;}
```

В данном примере создается два блока с разными характеристиками. В первом блоке вокруг текста по вертикали и горизонтали с помощью свойства **padding** задается одинаковое поле со значением 20 пикселей. Во втором блоке поле слева увеличено через свойство *padding-left*.

Установка высоты и ширины элемента

Установить высоту и ширину элемента можно с помощью свойств **height** и **width** соответственно. Однако при применении данных свойств существуют некоторые особенности. Например, данные свойства не могут применяться к *строковым элементам* HTML, таким, как, например, **SPAN**, **STRONG** или **EM**.

Работа с потоком элементов

Типы блоков: свойство display

Каждый элемент в Рекомендациях HTML 4.01, который связан с основным контентом, имеет соответствующий строковый или блочный тип. Каждый тип определяет поведение компоновки по умолчанию различным образом. Например, последовательно идущие строковые элементы изображаются на общей базовой линии, в то время как блочные элементы всегда отделяются друг от друга и выводятся с предшествующим и последующим разрывом строки.

Свойство **display** имеет три наиболее часто используемых значения - **block**, **inline** и **none** - два из которых имеют прямое отношение к соответствующим типам элементов. Данное свойство позволяет изменить поведение элементов (например, строковый элемент будет вести себя как блочный или наоборот). Свойство **display** со значением **none** может изменять представление данного элемента в документе. Например, с помощью следующего правила можно удалить посторонний фрагмент из заголовка:

```
.sectionNote {display: none;}
```

"Всплывающие" элементы: свойства float и clear

Элемент может "всплывать" вправо или влево с помощью свойства **float**. То есть бокс с его содержимым может сместиться к правому или левому краю в окне документа (или содержащего бокса). Если необходимо, например, чтобы текст окружал рисунок, как показано на [рисунке 13.4](#), то фрагмент кода должен выглядеть следующим образом:

```
...  
<STYLE type="text/css">  
#picture {  
float:left;
```

```

width: 130px;}
</STYLE>
...
<DIV id="picture">
<IMG src="Beatles.jpg" alt="The Beatles">
</DIV>
<P>Far away, 80000 leagues below the sea, ... </P>
...

```

Чтобы рисунок смещался влево, а текст его окружал, необходимо определить ширину бокса, окружающего рисунок, и установить в свойстве **float** значение **left**.

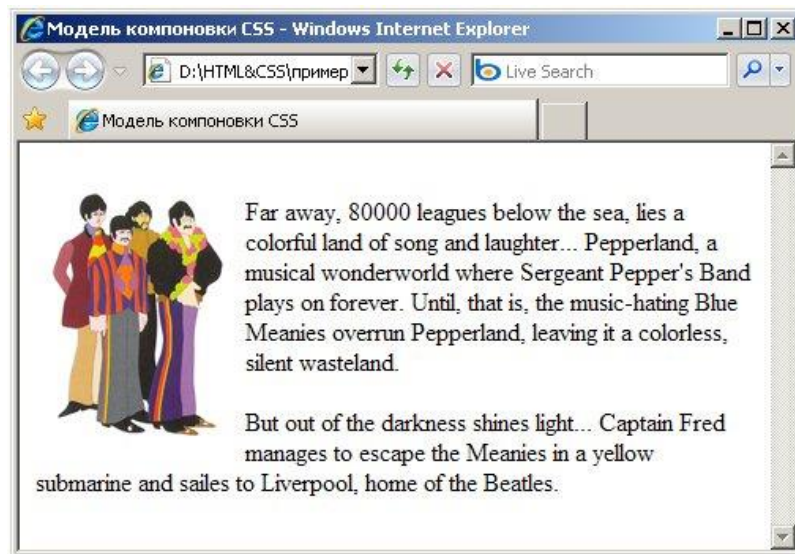


Рис. 16.4. Пример обтекания рисунка текстом

Свойство **clear** управляет поведением последовательности всплывающих элементов документа. По умолчанию, последовательные элементы смещаются вверх, заполняя доступное пространство, которое освобождается, если бокс смещается в сторону. Например, в предыдущем примере текст автоматически смещается вверх вдоль изображения. Свойство **clear** может иметь значения **left**, **right**, **both** или **none**.

Задание

1. К созданному ранее учебному сайту применить свойства, описанные выше.

Практическое занятие №17 Оформление ссылок и списков CSS

Цель занятия: формировать навык при работе с ссылками и списками.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Оформление списков

Базовые маркеры и числа: свойство `list-style-type`

Для управления видом маркера используется свойство *list-style-type*. Данное свойство применяется и к маркированному (элемент HTML UL) и нумерованному (элемент HTML OL) спискам, однако аргументы для этих двух видов списка различаются. Для маркированного списка используются аргументы *circle*, *disc* и *square*, которые устанавливают маркер в виде незакрашенного кружка, закрашенного кружка и квадрата соответственно. Для нумерованного списка аргументов свойства *list-style-type* намного больше, и с ними можно самостоятельно ознакомиться в Спецификации CSS2.1. На [рисунке 14.1](#) представлены несколько распространенных типов маркированного и нумерованного списков. Аргумент *none* устанавливает тип маркера, как у родительского элемента. По умолчанию данное свойство принимает значение *disc* для маркированного списка и *decimal* для нумерованного списка.

Например, следующие правила задают для всех маркированных списков на сайте квадратные маркеры, а для нумерованных - десятичные числа:

```
ul li {list-style-type: square;}  
ol li {list-style-type: decimal;}
```

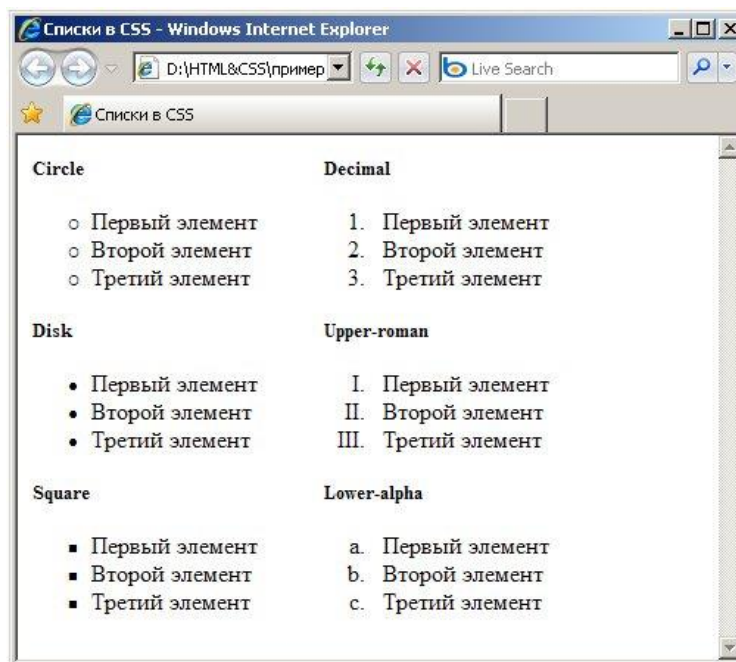


Рис. 17.1. Распространенные стили списков

Маркеры-изображения: свойство `list-style-image`

Хотя количество значений атрибута свойства `list-style-type` для элемента `UL`, т.е. для маркированного списка, ограничено тремя, это не значит, что в распоряжении разработчика или дизайнера всего три вида маркера. CSS позволяет установить в качестве маркера любое подходящее изображение с помощью свойства `list-style-image`. В качестве аргумента используется относительный или абсолютный адрес графического файла, содержащего изображение, которое должно служить в качестве маркера. В следующем примере для каждого элемента списка в качестве маркера устанавливается изображение `marker.png`:

```
UL {list-style-image: url("marker.png");}
```

Этот атрибут наследуется, поэтому для отдельных элементов списка для восстановления первоначально вида маркера используется значение атрибута `none`, которое отменяет изображение в качестве маркера для родительского элемента.

Следует заметить, что это свойство имеет ограниченные возможности позиционирования для фонового изображения и в некоторых ситуациях вообще не работает в браузере Internet Explorer. Поэтому значительно более распространенной практикой является просто задание фонового изображения в пунктах списка.

Прежде всего, необходимо определить для списка отсутствие маркера и удалить поле и заполнение:

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

Затем можно добавить фоновое изображение для каждого пункта списка, некоторое заполнение слева и снизу, чтобы сместить текст, позволяя вывести изображение, и растянуть пространство между пунктами списка:

```
UL LI {  
    background: #fff url("icon.gif") 0 3px no-repeat;  
    padding: 0 0 5px 15px;  
}
```

Размещение маркера: свойство `list-style-position`

Существует два способа размещения маркера относительно текста: маркер выносится за границу элементов списка или обтекает текст. Чтобы управлять положением маркера относительно текста, применяется свойство `list-style-position`. Это свойство имеет два значения: `outside` и `inside`. Значение `outside` (значение по умолчанию) размещает маркеры за пределами текстового блока. Значение `inside` включает маркеры в текстовый блок и отображает их в элементе списка. Пример использования этих значений показан на [рисунке 17.2](#).

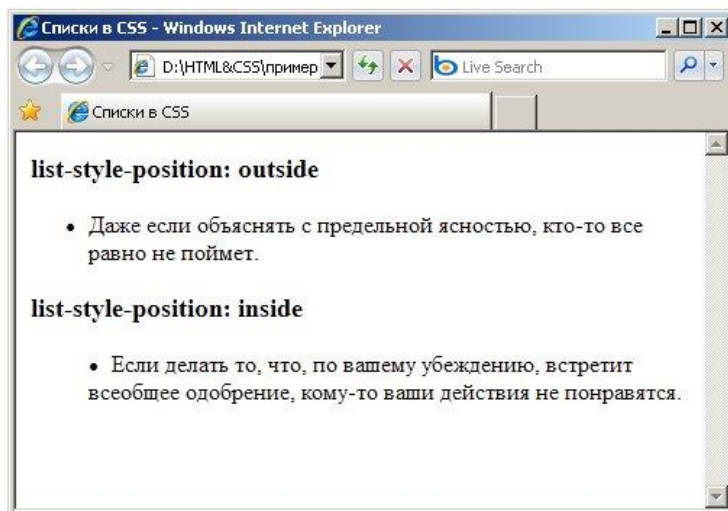


Рис. 17.2. Варианты размещения маркера

Оформление списков определений

Обычно списки определений не требуют большого внимания, за исключением задания стиля DT (обычно жирный текст) и управления отступом определений:

```
DT {font-weight: bold;}  
DD {margin-left: 2em;}
```

В CSS предусмотрена возможность одновременно задать стиль маркера, его положение, а также изображение, которое будет использоваться в качестве маркера. Для этой цели используется свойство *list-style*. В качестве аргументов данного свойства могут выступать любые комбинации трех значений, определяющих стиль маркеров, в произвольном порядке. Значения разделяются между собой пробелом. Ни один аргумент не является обязательным, поэтому неиспользуемые значения можно опустить.

Оформление ссылок

Существует несколько общих правил, которые разработчик должен учитывать при создании веб-страниц. Эти правила основываются на ожиданиях пользователей относительно оформления и действия ссылок:

- пользователи ожидают, что ссылки отличаются от остального текста, представленного на веб-странице, и что ссылкой является именно подчеркнутый текст;
- пользователи ожидают, что ссылки реагируют при наведении на них курсора и видоизменяются после того, как их посетили.

Таким образом, при оформлении ссылок разработчик должен задавать оформление для всех состояний ссылок и использовать подчеркивание только для ссылок

Оформление состояния ссылок

Стили ссылок всегда должны быть заданы в таблице стилей в следующем порядке: *link*, *visited*, *focus*, *hover* и *active*. Если стили ссылок будут размещены в другом порядке, то настройки будут переопределять друг друга, и состояния ссылок не будут работать.

Для оформления различных состояний ссылок используются *псевдоклассы* `:link`, `:visited`, `:focus`, `:hover` и `:active`, которые добавляют к селектору элемента `A`:

```
A:link{  
A:visited{  
A:focus{  
A:hover{  
A:active{
```

Если необходимо задать оформление для всех ссылок во всех состояниях, то можно оформлять непосредственно элемент `A`. Однако базовое правило должно быть определено в первую очередь:

```
A {  
A:link{  
A:visited{  
A:focus{  
A:hover{  
A:active{
```

Такая запись полезна, если необходимо убрать используемое по умолчанию подчеркивание ссылок.

Управление поведением по умолчанию

По умолчанию, большинство браузеров задает для всех ссылок подчеркивание, а состояние фокуса клавиатуры создает вокруг ссылок рамку. Данное оформление можно заменить или вообще отключить.

Подчеркивание задается с помощью свойства `text-decoration`. Напомним, что подчеркивание задается с помощью значения свойства `text-decoration`, равного `underline`:

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

Можно отключить подчеркивание с помощью следующего правила:

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

Установленное по умолчанию подчеркивание является толстоватым и пересекает нижние выносные элементы строчных букв. Если необходимо сохранить стиль подчеркивания ссылок, но сделать подчеркивание тоньше и запретить пересечение нижних выносных элементов, можно использовать ложное подчеркивание.

Прежде чем создавать ложное подчеркивание, необходимо отключить подчеркивание всех состояний ссылок:

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

Отключив заданное по умолчанию подчеркивание, можно задать свое подчеркивание с помощью свойства `border-bottom`:

```
A:link {border-bottom: 1px solid #00c;}
```

Результат применения описанных выше свойств к состоянию ссылки представлен на рисунке 17.3. Для сравнения представлена также ссылка, оформленная по умолчанию.



Рис. 17.3. Ложное подчеркивание в действии

При использовании метода ложного подчеркивания необходимо следить за тем, чтобы было задано достаточно большое значение **line-height**, чтобы избежать наложения подчеркивания на следующую строку текста.

Ложное подчеркивание позволяет создавать дизайн, в котором состояния ссылок можно отличать не только по цветам. Задавая различный стиль подчеркивания, можно гарантировать, что пользователь сможет различить состояния ссылок даже в черно-белом представлении.

Изображения возле ссылок

Некоторые сайты используют изображения и символы для добавления информации о своих ссылках. Например, можно использовать стрелку для указания, что ссылка позволяет перейти на внешний сайт, или применить какой-либо символ, чтобы отметить посещенные ссылки. Такие эффекты легко создаются с помощью фоновых изображений.

Чтобы добавить изображение к внешним ссылкам, вначале необходимо определить принадлежность такой ссылки к некоторому классу, в приведенном ниже примере это класс `external`:

```
<A href="http://www.somewhere.com /"
class="external">external link</A>
```

Затем необходимо задать фоновое изображение для этого класса:

```
A.external {
background: #fff url("arrow.gif") center right no-repeat;
padding-right: 30px;
}
```


Этот пример будет применять выбранное изображение ко всем экземплярам посещенных ссылок во всех состояниях. Если необходимо выделять с помощью изображений только непосещенные внешние ссылки, то можно объединить классы и *псевдоклассы* состояний ссылок следующим образом:

```
A.external:link{
  background: #fff url("arrow.gif") center right no-repeat;
  padding-right: 30px;
}
```

Объединение классов и состояний открывает широкие возможности для ссылок, в чем слушателю предлагается убедиться самостоятельно.

Задание

Предлагается средствами *CSS* установить базовые маркеры для нумерованных и маркированных списков, а также специальные маркеры, использующие графические изображения. Также рекомендуется создать горизонтальный *список* и *список* без маркеров.

Используя *псевдоклассы*, предлагается оформить различные состояния ссылок с помощью различного вида подчеркиваний, а также установить индивидуальные цвета. Необходимо установить для ссылок, указывающих на ресурсы Сети, определенное графическое изображение и предусмотреть смену изображения для уже посещенной ссылки. Также необходимо предусмотреть изменения вида ссылки при наведении на нее курсора: сделать ссылку перечеркнутой и изменить *цвет фона* ссылки.

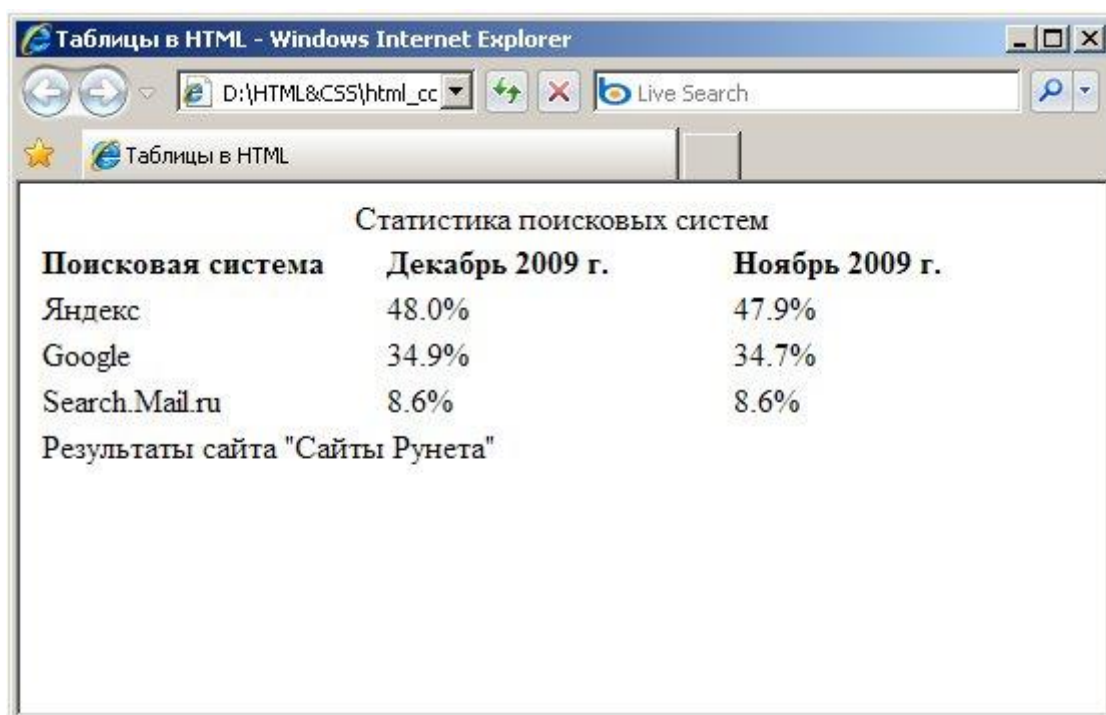
Практическая работа №18 Создание таблиц на сайте

Цель занятия: формировать навык при оформлении таблиц сайта с помощью CSS.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Таблицы обеспечивают возможность расположения многомерных данных по строкам и столбцам. Структура и содержание таблицы описываются с помощью элементов *HTML*, а ее оформление задается с помощью правил *CSS*. Визуальное оформление таблиц состоит в задании правил отображения заголовков и ячеек таблицы, их выравнивания относительно друг друга, изображения рамок вокруг них и т.д. Далее будут рассмотрены основные приемы визуального оформления таблиц.

Для дальнейшего изложения будет использоваться следующая нестилизованная *таблица*, представленная на [рисунке 18.1](#).



The screenshot shows a window titled "Таблицы в HTML - Windows Internet Explorer". The address bar contains "D:\HTML&CSS\html_cc". The page content is a table with the following data:

Статистика поисковых систем		
Поисковая система	Декабрь 2009 г.	Ноябрь 2009 г.
Яндекс	48.0%	47.9%
Google	34.9%	34.7%
Search.Mail.ru	8.6%	8.6%
Результаты сайта "Сайты Рунета"		

Рис. 18.1. Нестилизованная таблица

Ширина таблицы и ячеек

Для определения ширины таблицы и ячеек используется свойство `width`, в качестве значения которого принимаются любые единицы длины, принятые в *CSS* (пиксели, дюймы, пункты и др.) При использовании процентной записи ширина элемента вычисляется в зависимости от ширины родительского элемента, либо окна браузера. По умолчанию, браузер использует настройку `TABLE {width: auto;}`, что приводит к выводу таблицы во всю ширину окна браузера.

Следующие правила задают ширину таблицы в 100% доступной ширины, и ширину ячеек таблицы по 33% для каждой:

```
TABLE {width: 100%;}  
TH, TD {width: 33%;}
```

Браузеры неодинаково работают с шириной, кроме того, результат отображения зависит от используемого **DOCTYPE**. Так, в Internet Explorer при использовании переходного **DOCTYPE** или при его отсутствии, если содержимое превышает заданную ширину, то блок изменяет свои размеры, подстраиваясь под содержимое. В противном случае *ширина блока* равна значению `width`. В то время, как для строгого **DOCTYPE** ширина формируется путем сложения значений `width`, `padding`, `margin` и `border`. Если содержимое блока не помещается в заданные размеры, оно отображается поверх.

Выравнивание в таблице

Часто необходимо менять установленные по умолчанию настройки выравнивания текста в таблице. CSS позволяет задавать выравнивание текста в таблице по горизонтали и вертикали. Для этой цели служат свойства `text-align` и `vertical-align`. Два данных свойства подробно описаны ранее, поэтому ниже приведен лишь пример применения данных свойств к оформлению текста в ячейках таблицы:

```
TH, TD {  
    width: 33%;  
    text-align: left;  
    vertical-align: top;  
}
```

Отображение границ

Чтобы четко отделить содержимое одной ячейки от другой, к ним добавляются границы. За создание границ отвечает атрибут `border` элемента **TABLE**, который определяет толщину рамки. Однако, рамки, созданные с помощью данного атрибута, получаются разными по своему виду в каждом браузере. Чтобы этого избежать, рекомендуется пользоваться CSS-свойством `border`, применяя его к таблице или ее ячейкам (элементам **TD** или **TH**).

Ниже приведен пример использования данных свойств для оформления таблиц:

```
TABLE {border: 1px solid #000;}  
TH, TD {border-left: 1px dashed #000;}
```

Поскольку границы создаются для каждой ячейки, то в местах соприкосновения ячеек получается граница удвоенной толщины. Самым простым способом устранения указанной особенности является применение свойства `border-collapse`, которое устанавливает, как именно отображать границы вокруг ячеек. Данное свойство имеет два значения: `collapse` и `separate`. Значение `collapse` заставляет браузер анализировать места соприкосновения ячеек в таблице и убирать в ней двойные линии. При этом между ячейками остается только одна граница, одновременно принадлежащая обоим ячейкам. То же правило соблюдается и для внешних границ, когда вокруг самой таблицы добавляется рамка. При использовании значения `separate`, которое устанавливается по умолчанию, вокруг каждой ячейки отображается своя собственная рамка, а в местах соприкосновения ячеек показываются сразу две линии.

Различия между двумя значениями свойства `border-collapse` представлены на [рисунке 18.2](#). Первая таблица соответствует границам удвоенной толщины и задается правилом

```
TABLE {  
    border: 1px solid #000;  
    border-collapse: separate;}
```

Вторая таблица соответствует схлопнувшимся границам и задается правилом

```
TABLE {  
border: 1px solid #000;  
border-collapse: collapse;}
```

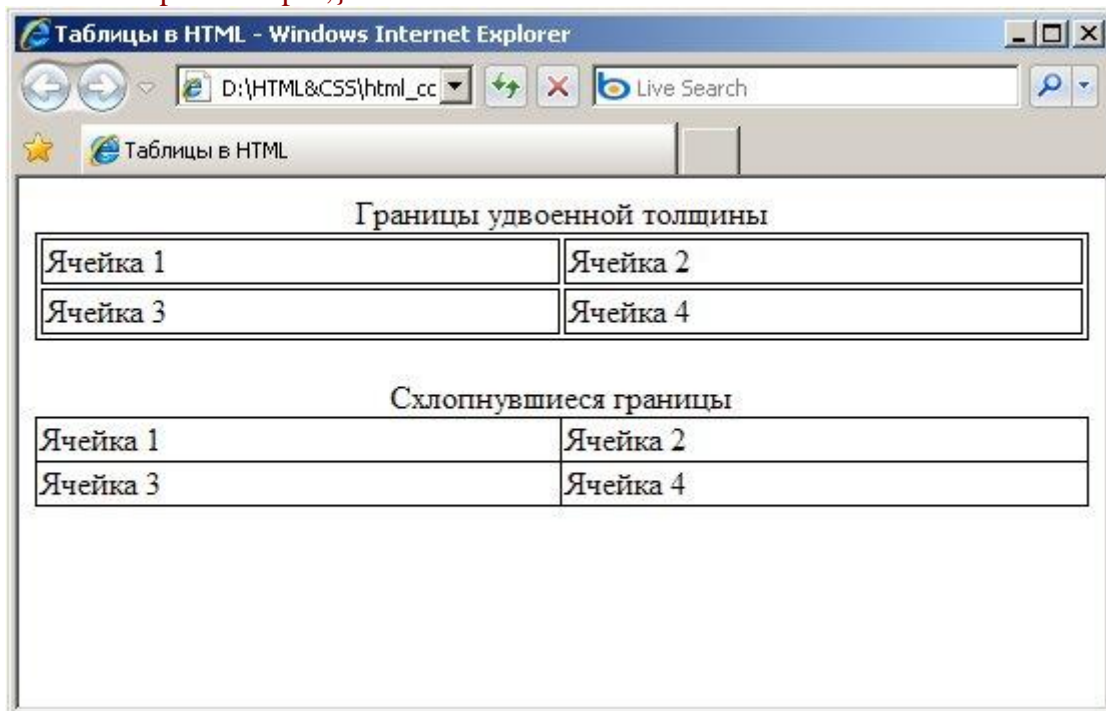


Рис. 18.2. Результат применения различных значений свойства border-collapse

Когда задается схлопывание границ, необходимо помнить, что это может создавать проблемы, если к границам смежных ячеек были применены различные *стили оформления*. Схлопывание границ с различными стилями приводит к конфликтам, которые разрешаются согласно *правилам разрешения конфликтов границ таблиц* спецификации CSS2 (<http://www.w3.org/TR/REC-CSS2/tables.html#border-conflict-resolution>). Данные правила определяют, какие стили "выигрывают" при схлопывании границ.

Расстояние между ячейками

CSS позволяет также увеличивать расстояния между ячейками, используя свойство *border-spacing*. Данное свойство задает расстояние между границами ячеек в таблице и не действует в случае, когда для таблицы установлен параметр *border-collapse* со значением *collapse*. Одно значение устанавливает одновременно расстояние по вертикали и горизонтали между границами ячеек. Если значений два, то первое определяет горизонтальное расстояние, а второе - вертикальное.

Следующие правило позволяет установить расстояние между границами ячеек 150 пикселей по горизонтали и 20 пикселей по вертикали:

```
TABLE {  
border-collapse: separate;  
border-spacing: 150px 20px;}
```

Браузер Internet Explorer до восьмой версии не обрабатывает свойство *border-spacing*, поэтому пользоваться данным свойством надо с осторожностью.

Заполнение

Для ячеек, которые имеют границу, можно добавить свободное пространство между границами ячеек и их содержимым. Для этого используется свойство **padding**.

Ниже приведен пример применения данного свойства к оформлению ячеек:

```
TH, TD {  
  border: 1px solid #000;  
  border-collapse: collapse;  
  padding: 0.3em;  
}
```

Размещение заголовка таблицы

По умолчанию, заголовок таблицы размещается сверху таблицы. Однако в некоторых браузерах возможно переместить заголовок таблицы в другое место с помощью свойства *caption-side*. Значениями данного свойства являются **top**, **bottom**, **left** и **right**, размещающие заголовок сверху, внизу, слева или справа таблицы. В некоторых браузерах, как, например, Internet Explorer, доступны только два значения **top** и **bottom**. Следующее правило поместить заголовок таблицы под ней:

```
CAPTION {caption-side: bottom;}
```

Аналогичного результата можно достичь, используя свойство

```
TABLE {caption-side: bottom;}
```

Шаблоны таблиц

К таблицам рекомендуется применять определенное оформление, что поможет не только вписать их в общий *дизайн* веб-страницы, но и представить информацию в более наглядном виде. Далее представлены несколько приемов оформления таблиц с помощью стилей.

Простой дизайн

Широко используемым вариантом дизайна для таблиц является выделение ячеек и строк с помощью фона. Так, в представленной ниже таблице заголовок таблицы выделен путем использования белого текста на темном фоне (так называемая выворотка), а для заголовков столбцов таблицы задан серый фон:

```
TABLE {  
  width: 100%;  
  border: 1px solid #000;  
  border-collapse: collapse;  
}
```

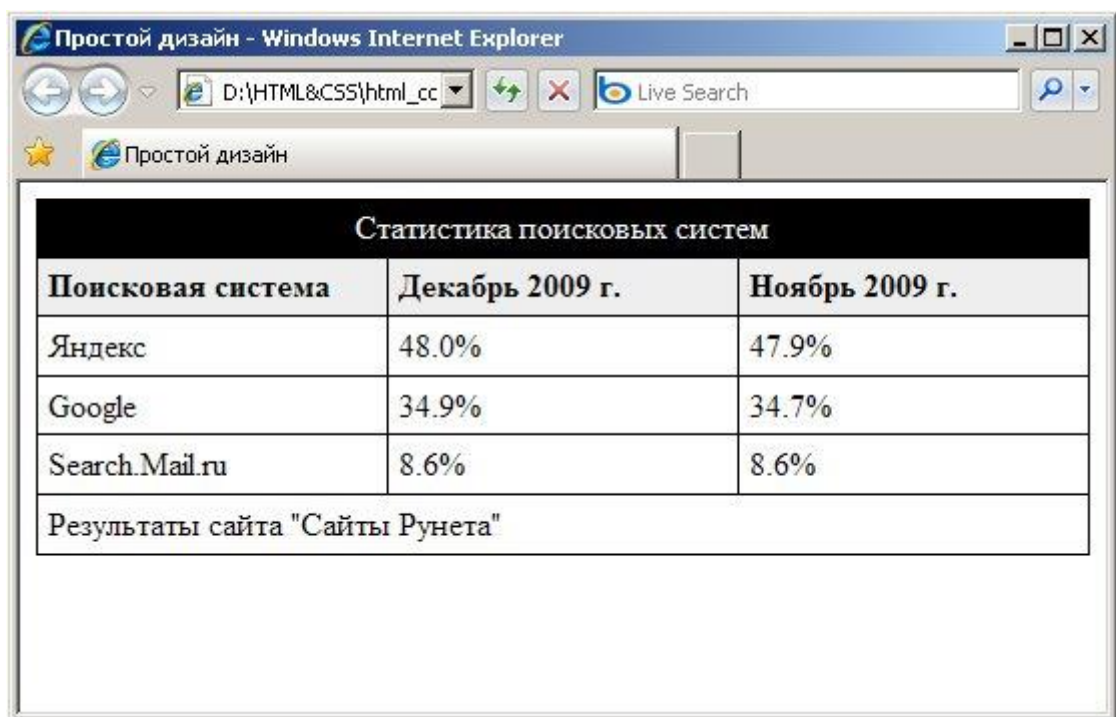
```
TH, TD {  
  width: 33%;  
  text-align: left;  
  vertical-align: top;  
  border: 1px solid #000;  
  padding: 0.3em;
```

```
caption-side: bottom;
}
```

```
CAPTION {
padding: 0.3em;
color: #fff;
background: #000;
}
```

```
TH {
background: #eee;
}
```

Полученный результат представлен на [рисунке 18.3](#).



The screenshot shows a Windows Internet Explorer browser window titled "Простой дизайн - Windows Internet Explorer". The address bar shows the file path "D:\HTML&CSS\html_cc" and a search engine "Live Search". The page content is a table with the following data:

Статистика поисковых систем		
Поисковая система	Декабрь 2009 г.	Ноябрь 2009 г.
Яндекс	48.0%	47.9%
Google	34.9%	34.7%
Search.Mail.ru	8.6%	8.6%
Результаты сайта "Сайты Рунета"		

Рис. 18.3. Таблица с простым дизайном

Разметка "зебры"

Для удобного представления данных в таблице можно сделать строки таблицы чередующимися, чтобы цвет фона четных и нечетных строк различался. Такая разметка обычно называется "зеброй". Хотя и существуют сомнения в отношении того, насколько данная разметка действительно облегчает восприятие информации, она является популярным *стилем оформления*. Пример использования данного оформления представлен на [рисунке 18.4](#).

Статистика поисковых систем		
Поисковая система	Декабрь 2009 г.	Ноябрь 2009 г.
Яндекс	48.0%	47.9%
Google	34.9%	34.7%
Search.Mail.ru	8.6%	8.6%
Результаты сайта "Сайты Рунета"		

Рис. 18.4. Таблица с разметкой "зебры"

Для изменения цвета фона у определенных строк прежде всего необходимо ввести классы для четных и нечетных строк:

```
...
<TABLE>
  <TR class="odd">
    ...
  <TR class="even">
    ...
</TABLE>
```

Затем необходимо добавить селектор для задания фона всех ячеек в строках, относящихся к заданным классам, например:

```
.odd th, .odd td {background: #eee;}
```

Неполные сетки

Для представления некоторых данных хорошо подходят менее структурированные таблицы. Простым вариантом является удаление вертикальных границ и заливки фона заголовка таблицы, как показано на [рисунке 18.5](#).

Поисковая система	Декабрь 2009 г.	Ноябрь 2009 г.
Яндекс	48.0%	47.9%
Google	34.9%	34.7%
Search.Mail.ru	8.6%	8.6%
Результаты сайта "Сайты Рунета"		

Рис. 18.5. Таблица с границами только на внешних краях и по нижнему краю каждой ячейки

Код CSS для этого представления может иметь следующий вид:

```
TABLE {
  width: 100%;
  border: 1px solid #999;
  text-align: left;
  border-collapse: collapse;
  margin: 0 0 1em 0;
  caption-side: top;
}

CAPTION, TD, TH {
  padding: 0.3em;
}

TH, TD {
  border-bottom: 1px solid #999;
  width: 25%;
}
```

Можно удалить все границы, за исключением верхней и нижней, чтобы определить только основное содержимое таблицы. Пример такого оформления представлен на [рисунке 18.6](#).

Таблицы в HTML - Windows Internet Explorer

D:\HTML&CSS\html_cc

Live Search

Таблицы в HTML

Статистика поисковых систем

Поисковая система	Декабрь 2009 г.	Ноябрь 2009 г.
Яндекс	48.0%	47.9%
Google	34.9%	34.7%
Search.Mail.ru	8.6%	8.6%

Результаты сайта "Сайты Рунета"

Рис. 18.6. Таблица с границами только вверху и внизу тела таблицы

Код CSS для такой таблицы будет следующим:

```
TABLE {
  width: 100%;
  text-align: left;
  border-collapse: collapse;
  margin: 0 0 1em 0;
  caption-side: top;
}

CAPTION, TD, TH {
  padding: 0.3em;
}

TBODY {
  border-top: 1px solid #000;
  border-bottom: 1px solid #000;
}

TBODY TH, TFOOT TH {
  border: 0;
}

TFOOT {
  text-align: center;
  color: #555;
  font-size: 0.8em;
}

```

Задание

Рекомендуется последовательно применить к созданной ранее таблице учебного сайта оформление, добавляя ко всей таблице и ячейкам индивидуальное оформление, включающее толщину и стиль границы, заполнение и выравнивание текста в ячейках, высоту и ширину и т.д.

Рекомендуется также применить к созданной таблице основные шаблоны оформлений, описанные в теоретическом материале.

Практическое занятие №19 Позиционирование в CSS

Цель занятия: формировать навык создания сайта с применением позиционирования CSS; управления размещением структурных блоков и их содержимого; проверка HTML-документов и CSS-кодов.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Прежде чем рассматривать *позиционирование*, в двух словах напомним, что с точки зрения боксовой модели каждый элемент *HTML* представляет собой *прямоугольник* (бокс), для которого можно задать такие параметры как поля, границы и заполнения. *Позиционирование* определяет, где должен располагаться этот *прямоугольник*, а также как он должен влиять на элементы вокруг себя. При помощи позиционирования можно разместить любой элемент точно в нужном месте страницы. Вместе со всплывающими элементами, рассмотренными ранее, *позиционирование* дает большие возможности для создания оригинального дизайна.

В основе позиционирования лежит *представление* окна браузера как системы координат. Любой бокс возможно расположить в этой системе координат где угодно. Например, представленное ниже правило позволяет расположить заголовок на расстоянии 100 пикселей от верхней границы документа и на 200 пикселей от левой границы документа:

```
H1 {  
  position: absolute;  
  top: 100px;  
  left: 200px;  
}
```

Результат выполнения данного кода представлен на [рисунке 19.1](#).

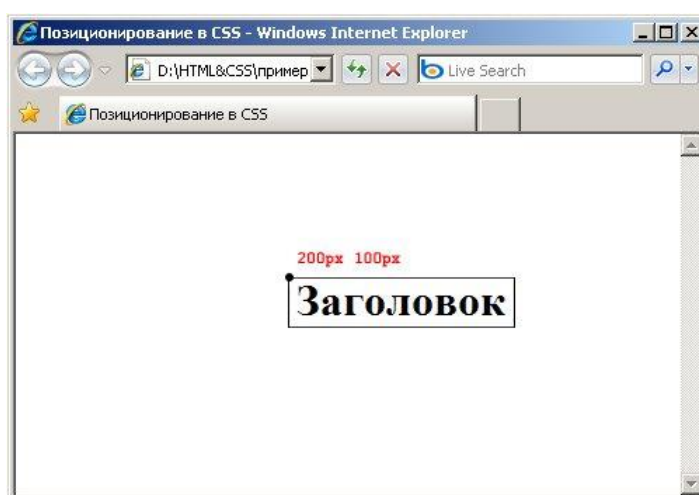


Рис. 19.1. Пример позиционирования заголовка

Из приведенного примера понятно, что для позиционирования элементов используется свойство **position**. Свойство **position** имеет четыре значения **static**, **relative**, **absolute** и **fixed**, которые определяют тип позиционирования и влияют на расположение элемента.

Статическое позиционирование

Значение **static** свойства **position** используется по умолчанию. Любой элемент со статическим позиционированием находится в общем потоке документа. Правила для его размещения определяются боксовой моделью. Блочные и *строковые элементы* размещаются по разным правилам, четко определенным в Спецификации CSS2.1.

По умолчанию, блочные боксы выкладываются вертикально сверху вниз в порядке появления их в разметке. Каждый бокс обычно занимает всю ширину документа и имеет разрыв строки перед и после себя. Вертикальное расстояние между двумя блочными боксами управляется свойством **margin-bottom** первого блока и свойством **margin-top** второго бокса, причем *вертикальные поля* между двумя последовательными блочными боксами будут перекрываться таким образом, что расстояние между ними будет определяться не суммой двух полей, а большим из них.

Строковые боксы выстраиваются по горизонтали в том порядке, в котором они появляются в разметке, переходя на новую строку, только если исчерпано доступное горизонтальное пространство. В зависимости от свойства **direction**, строковые боксы будут располагаться либо слева направо (**direction: ltr**), либо справа налево (**direction: rtl**).

Множество строковых боксов, которые составляют строку на экране, заключаются еще в один прямоугольник, называемый линейным боксом. Линейные боксы выкладываются вертикально в своих пределах блочного уровня без дополнительного пробела между ними. Высотой линейных боксов можно управлять с помощью свойства **line-height**. Для строковых боксов нельзя определить размеры и *вертикальные поля*.

Относительное позиционирование

Элемент со значением свойства **position**, равным **relative**, сначала размещается по правилам статического позиционирования. Но затем сгенерированный бокс смещается относительно своего положения в потоке, согласно значениям свойств **top**, **bottom**, **left** и **right**. Но при этом из потока он не исключается, а продолжает занимать там свое место. То есть сдвигается со своего места он только визуально, а положение всех боксов вокруг него никак не меняется. Это означает, что смещенный бокс может перекрывать боксы других элементов, так как они по-прежнему действуют, как если бы относительно позиционированный элемент остался там, где он должен был быть перед применением позиционирования. Как пример *относительного позиционирования*, попробуем сместить текст, заключенный в элемент `...` относительно его начального положения на странице с помощью следующего фрагмента кода:

```
SPAN {  
  position: relative;  
  top: 10px;  
  left: 10px;  
  background-color: lime;  
}
```

...

`<P>` В представленном ``фрагменте`` текста некоторые слова находятся не на своем месте и перекрывают другие слова.`</P>`

Результат выполнения данного кода представлен на [рисунке 19.2](#). Блок теперь перекрывает следующую строку текста, а на его прежнем месте появилось пустое пространство.

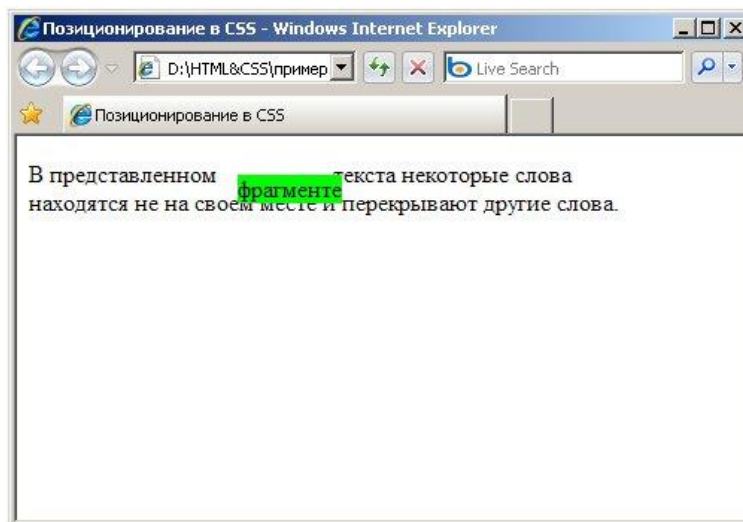


Рис. 19.2. Пример относительно позиционирования
Абсолютное и фиксированное позиционирование

Бокс с абсолютным позиционированием располагается по заданным координатам, а из того места, где он должен был бы быть, он удаляется, и в этом месте сразу начинают размещаться следующие боксы. Считается, что бокс исключается из потока.

Для *абсолютного позиционирования элемента* свойство `position` должно принимать значение **absolute**. А для задания положения размещения блока используются значения **left**, **right**, **top** и **bottom**. Все четыре свойства можно использовать одновременно для определения расстояния от каждого края позиционируемого элемента до соответствующего края браузера или родительского блока. Можно определить также позицию одного из углов абсолютно позиционируемого (например, используя **top** и **left**), а затем определить размеры бокса, используя **width** и **height**.

Представленный ниже код позволяет разместить четыре бокса в разных углах HTML-документа:

```
div#yellow1 {  
    position:absolute;  
    top: 10px;  
    left: 10px;  
}
```

```
div# yellow2 {  
    position:absolute;  
    top: 50px;  
    right: 50px;  
}
```

```
div# yellow3 {  
    position:absolute;  
    bottom: 10px;  
    right: 10px;  
}
```

```
div#yellow4 {
```

```
position:absolute;
bottom: 10px;
left: 10px;
}
```

Результат применения описанных выше свойств представлен на [рисунке 19.3](#).

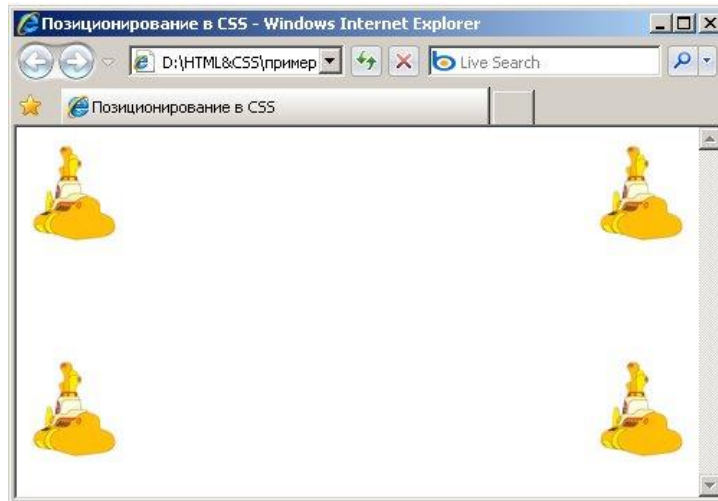


Рис. 19.3. Пример абсолютного позиционирования

Фиксированное позиционирование действует подобно абсолютному, однако элемент с фиксированным позиционированием всегда располагается только относительно окна браузера и никогда не смещается при прокручивании веб-страницы. Отметим, что Internet Explorer версии 6 и более ранних версий не поддерживает фиксированное позиционирование.

Третье измерение веб-страницы

Страница сайта двумерна: для нее заданы ширина и высота. CSS позволяет добавить к веб-странице глубину (третье измерение) с помощью свойства **z-index**. Данное свойство позволяет создавать слои и располагать одни элементы поверх других.

Для создания слоев необходимо для каждого элемента задать значение свойства **z-index**, которое является своеобразным порядковым номером слоя, в котором находится данный элемент. Это значение может быть целым числом (которое может быть отрицательным) или одним из ключевых слов **auto** или **inherit**. Значением по умолчанию является **auto** или **0**. Элемент с большим значением свойства **z-index** перекрывает элемент с меньшим значением данного свойства. Ниже представлен код, располагающий игральные карты в порядке возрастания. Результат выполнения данного кода представлен на [рисунке 19.4](#).

```
div#card1 {
    position:absolute;
    top: 50px;
    left: 150px;
    z-index: 1;
}
```

```
div#card2 {
    position:absolute;
    top: 70px;
```

```

left: 170px;
z-index: 2;
}

div#card3 {
position: absolute;
top: 90px;
left: 190px;
z-index: 3;
}

```

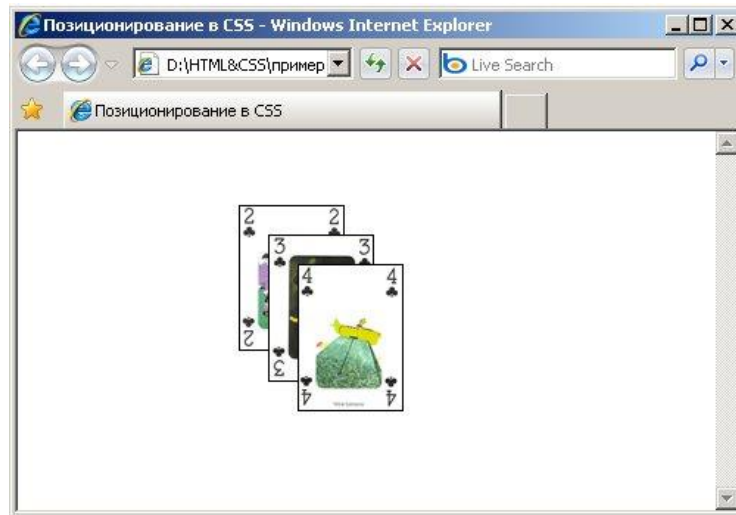


Рис. 19.4. Пример использования свойства z-index

Задание

1. Необходимо, используя свойства, управляющие компоновкой и позиционированием элементов, облагородить вид созданных веб-страниц: установить расстояния между блоками, оформить их границами, установить высоту и ширину блоков, обтекание графических изображений на странице текстом, используя "всплывающие" элементы, и т.д. С помощью различных типов позиционирования предлагается реализовать виды разметки веб-страниц, представленных на рисунке 19.5.

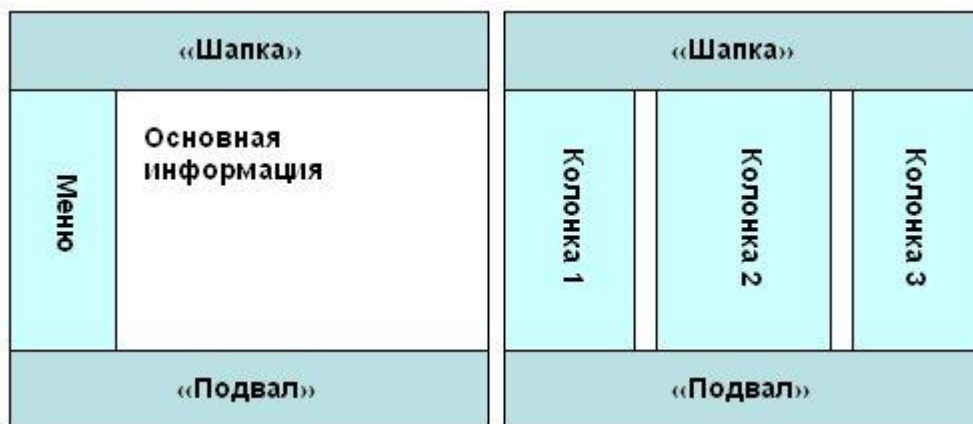


Рис. 19.5. Разметка веб-страниц

2. Проверить, созданные на предыдущих практических занятиях веб-страницы, с помощью валидаторов W3C Markup Validator и W3C CSS Validator. В случае обнаружения ошибок, разобрать их и исправить.

Тематика практических и самостоятельных работ по разделу 4 HTML 5

Тема работы	Вид работы	Содержание занятия	Количество часов
Применение HTML5 для разработки индивидуально сайта	Практическое занятие	Занятие № 20-21 Применение HTML5 для разработки индивидуально сайта	4
Применение HTML5 для разработки индивидуально сайта	Самостоятельная работа		10

Практическое занятие №20-21 Применение HTML5 для разработки индивидуального сайта

Цель занятия: формировать навык по разработке сайта с применением тегов HTML5.

Задание:

1. Добавить на сайт видео.
2. Применить API геолокацию.
3. Добавить новые элементы управления форм.
4. Добавить структурные элементы HTML5:
 - **<header>**: Используется для верхнего колонтитула сайта.
 - **<footer>**: Используется для нижнего колонтитула сайта.
 - **<nav>**: Содержит навигационные функции страницы.
 - **<article>**: Содержит автономный фрагмент контента, который будет иметь смысл, если используется как позиция RSS, например, новостное сообщение.
 - **<section>**: Используется либо для объединения в группу различных статей с различной целью или по различным темам, или для определения различных разделов одной статьи.
 - **<time>**: Используется для разметки времени и даты.
 - **<aside>**: Определяет блок контента, который связан с основным контентом, но не входит в его основной поток.
 - **<hgroup>**: Используется в качестве оболочки скрытия более одного заголовка, если требуется, чтобы учитывался только один заголовок в структуре заголовков страницы.
 - **<figure>** и **<figcaption>**: Используется для инкапсуляции рисунка как единого элемента, и содержит, соответственно, подпись для рисунка.
5. Применить элемент canvas.

Тематика практических и самостоятельных работ по разделу 5 Java Script

Тема работы	Вид работы	Содержание занятия	Количество часов
Процедуры и функции	Практическое занятие	Практическое занятие №22-23 Разработка скриптов по созданию функций	4
Математика на веб-страницах	Практическое занятие	Практическое занятие №24 Разработка скриптов для решения квадратного уравнения, табулирования функции, распределения чисел по свойствам (четное, нечетное, простое), перевода единиц (из одной системы счисления в другую, из градусов по Цельсию в градусы по Кельвину, из дюймов в сантиметры и др.), формирования корзины покупателя; разработка сценария, который генерирует последовательность из N чисел, отражающую ход кривой дракона, то есть очередность поворотов кривой.	2
Создание скриптов	Самостоятельная работа	Выполнить задания для самостоятельной работы в данном разделе	10

Практическое занятие №22-23 Процедуры и функции

Цель занятия: формирование умений и навыков при разработке сценариев по созданию функций для решения различных математических задач, для рисования секторов, для решения банковских задач.

Ход работы

Выполнить задания № 1, 3, 5, 7, 9, 11, 13, 15 (стр. 49-51).

Учебник: Диков А.В Web-программирование на JavaScript: учебное пособие для СПО/ А.В. Диков. – 2-е изд., стер. – Санкт-Петербург: Лань, 2022.

Практическое занятие №24 Математика на веб-страницах

Цель занятия: формирование умений и навыков по применению команд JavaScript для решения математических задач.

Ход работы

Выполнить задания № 1-7 (стр. 60-63).

Учебник: Диков А.В Web-программирование на JavaScript: учебное пособие для СПО/ А.В. Диков. – 2-е изд., стер. – Санкт-Петербург: Лань, 2022.

Самостоятельная работа

Java Script

Цель: формирование навыков по созданию сценариев JavaScript.

Часть 1 Введение в JavaScript. Программное взаимодействие с HTML документами на основе DOM API.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Элементы языка JavaScript

JavaScript позволяет "оживить" веб-страницу. Это реализуется путем добавления к статическому описанию фрагмента исполняемого кода. JavaScript-сценарий может взаимодействовать с любыми компонентами HTML-документа и реагировать на изменение их состояния.

JavaScript не является строго типизированным языком, в переменных могут храниться практически любые типы данных.

Как и программа на языке Java, сценарий JavaScript выполняется под управлением интерпретатора. Однако если Java-приложение или Java-апплет компилируется в байтовый код, то сценарий JavaScript интерпретируется на уровне исходного текста.

Следует отметить, что языковые конструкции JavaScript совпадают с соответствующими средствами C++ и Java.

Структура сценария

Сценарием JavaScript считается фрагмент кода, расположенный между дескрипторами `<SCRIPT>` и `</SCRIPT>`:

```
Текст HTML-документа
<SCRIPT>
  Код сценария
</SCRIPT>
Текст HTML-документа
```

Переменные

В сценариях JavaScript переменные могут хранить данные любых типов: числа, строки текста, логические значения, ссылки на объекты, а также специальные величины, например "нулевое" значение `null` или значение `NaN`, которое сообщает о недопустимости операции.

Переменная в языке JavaScript объявляется с помощью ключевого слова `var`. Так, например, выражение

```
var selected = "first item";
```

создает переменную с именем `selected` и присваивает ей в качестве значения строку символов `"first item"`. Переменные могут объявляться также автоматически. Это происходит при присвоении значения переменной, не встречавшейся ранее в данном сценарии. Так, в следующем примере создается переменная с именем `rating`, которой присваивается числовое значение, равное `512.5`:

```
rating = 512.5;
```

Объекты

В языке JavaScript не предусмотрены средства для работы с классами в том виде, в котором они реализованы в C++ или Java. Разработчик сценария не может создать подкласс на основе существующего класса, переопределить метод или выполнить какую-либо другую операцию с классом. Сценарию, написанному на языке JavaScript, в основном доступны лишь готовые объекты. Построение нового объекта приходится выполнять лишь в редких случаях.

Объекты содержат свойства (свойства объектов можно сравнить с переменными) и методы. Объекты, а также их свойства и методы идентифицируются именами. Объектами являются формы, изображения, гипертекстовые ссылки и другие компоненты веб-страницы, HTML-документ, отображаемый в окне браузера, окно браузера и даже сам браузер. В процессе работы JavaScript сценарий обращается к этим объектам, получает информацию и управляет ими.

Кроме того, разработчику сценария на языке JavaScript доступны объекты, не связанные непосредственно с HTML-документом. Их называют предопределенными, или независимыми объектами. С помощью этих объектов можно реализовать массив, описать дату и время, выполнить математические вычисления и решить некоторые другие задачи.

Первый объект, с которым необходимо познакомиться, чтобы написать простейший сценарий, - это объект `document`, который описывает HTML документ, отображаемый в окне браузера. Ниже приведен исходный текст веб-страницы, содержащей сценарий, действия которого сводятся к выводу строки текста в окне браузера.

```
<HTML>
  <HEAD>
    <TITLE>Первый сценарий JavaScript</TITLE>
  </HEAD>
  <BODY>
    <SCRIPT LANGUAGE="JavaScript">
      document.write("Проверка сценария JavaScript");
    </SCRIPT>
  </BODY>
</HTML>
```

Имена чувствительны к регистрам символов, и если вы попытаетесь обратиться к текущему документу по имени `Document`, интерпретатор JavaScript отобразит сообщение об ошибке.

Основное назначение сценариев JavaScript - создавать динамически изменяющиеся объекты, корректировать содержимое HTML-документов в зависимости от особенностей окружения, осуществлять взаимодействие с пользователем и т.д.

Операции

Набор операторов в JavaScript, их назначение и правила использования в основном совпадают с принятыми в языке C++. Исключением является операция задаваемая символом "+".

В JavaScript символ "+" определяет как суммирование числовых значений, так и конкатенацию строк.

Так, например, в результате вычисления выражения

```
sum = 47 + 21;
```

переменной `sum` будет присвоено значение 68, а после выполнения операции

```
sum = "строка 1 " + "строка 2";
```

в переменную `sum` будет записана последовательность символов "строка 1 строка 2".

Рассмотрим еще один пример:

```
<HTML>
  <BODY>
    <H2>Числа и строки</H2><BR>
    <SCRIPT LANGUAGE="JavaScript">
```

```

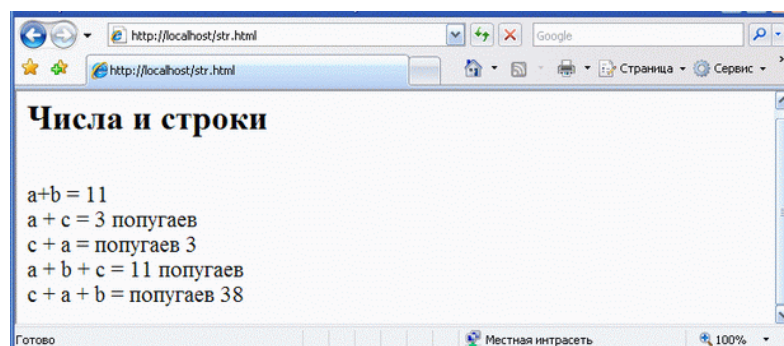
var a = 3;
var b = 8;
var c = " попугаев ";

document.write("a+b="); document.write(a + b);
document.write("<BR>");
document.write( "a + c = "); document.write(a+c);
document.write("<BR>");
document.write("c + a = "); document.write ( c + a);
document.write ("<BR>");
document.write ("a + b + c = "); document.write(a + b + c);
document.write("<BR>");
document.write("c + a + b = "); document.write(c + a + b);
document.write("<BR>");

</SCRIPT>
</BODY>
</HTML>

```

В окне браузера приведенный выше HTML-код выглядит так, как показано на скриншоте



Первая строка отображает результат суммирования двух числовых значений, вторая и третья - результат конкатенации строки и символического представления числа. Если операция суммирования чисел предшествует конкатенации, JavaScript вычисляет сумму чисел, представляет ее в символическом виде, затем производит конкатенацию двух строк. Если же первой в выражении указана операция конкатенации, то JavaScript сначала преобразует числовые значения в символический вид, а затем выполняет конкатенацию строк.

Управляющие конструкции

Управляющие конструкции, используемые в языке C++, в основном применимы и в сценариях JavaScript.

В JavaScript дополнительно определены языковые конструкции, отсутствующие в C++, а именно: операторы `for...in` и `with`.

В примере с помощью оператора цикла на веб-странице формируется таблица умножения чисел.

```

<html>
<body>
<table>
<script language="JavaScript">
  document.write("<tr><td> </td>");
  for (i = 1; i < 10; i++) document.write("<td>"+i+" </td>");
  document.write("</tr>");
  for (i = 1; i < 10; i++)
  {
    document.write("<tr><td>" + i + " </td>");
    for (j = 1; j < 10; j++)
    {

```

```

        document.write("<td        bgcolor='#00ffa0'>" + (i*j) +
" </td>");}
        document.write("</tr>");
    }

</script>
</table>
</body>
</html>

```

Отдельного внимания заслуживает оператор `new`. Несмотря на то, что большинство объектов уже созданы браузером и доступны сценарию, в некоторых случаях приходится создавать объекты в процессе работы. Это относится к предопределенным объектам и объектам, определяемым разработчиком сценария. Для создания объекта используется оператор `new`, который вызывается следующим образом:

```
переменная = new тип_объекта (параметры)
```

Функции

Формат объявления функции выглядит следующим образом:

```
function имя_функции ([ параметры]) тело_функции
```

Объявление функции начинается с ключевого слова `function`. Так же, как и в языке *С* для идентификации функции используется имя, при вызове функции могут передаваться параметры, а по окончании выполнения возвращаться значение. Однако, в отличие от *С*, тип возвращаемого значения и типы параметров не задаются. Ниже показаны два способа вызова функции

- `имя_функции ([параметры]);`
- `переменная = имя_функции ([параметры]);`

Во втором случае значение, возвращаемое функцией, присваивается указанной переменной.

Область видимости переменных

Работа с переменными в теле функции подчиняется следующим правилам.

❑ Если переменная объявлена с помощью ключевого слова `var`, доступ к ней осуществляется по правилам, подобным тем, которые используются в языке *С*.

❑ Переменная, объявленная внутри функции, считается *локальной*. Область видимости такой переменной ограничивается телом функции, в которой она объявлена.

❑ Переменная, объявленная вне функции, считается *глобальной*. К ней можно обращаться из любой точки сценария.

- Если локальная и глобальная переменные имеют одинаковые имена, то в теле функции локальная переменная "маскирует" глобальную.

Если переменная создается автоматически, т.е. если она не объявлена с помощью ключевого слова `var`, но присутствует в левой части оператора прямого присваивания, то она считается глобальной и становится доступной из любой точки сценария.

HTML DOM

DOM (Document Object Model) – представляет собой стандарт консорциума W3C для программного доступа к документам HTML или XML. Фактически это платформи- и языково-нейтральный интерфейс, позволяющий программам и сценариям динамически обращаться и обновлять содержимое, структуру и стиль документа.

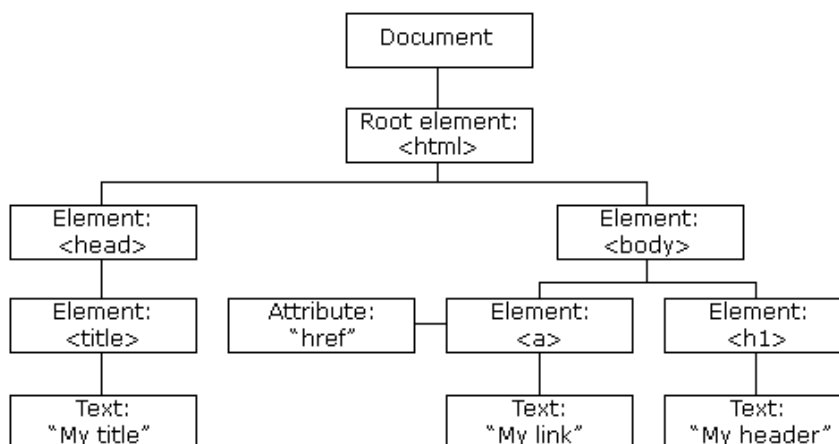
В рамках данного стандарта можно выделить 3 части:

- Core DOM – стандартная модель любого структурированного документа
- XML DOM - стандартная модель XML документа
- HTML DOM - стандартная модель HTML документа

DOM определяет объекты и свойства всех элементов документа и методы (интерфейс) для доступа к ним.

HTML DOM определяет объекты и свойства всех HTML элементов и методы (интерфейс) для доступа к ним. Иначе говоря, HTML DOM описывает каким образом необходимо получать, изменять, добавлять и удалять HTML-элементы.

В соответствии с моделью DOM все, что содержится внутри HTML документа - является узлом. То есть HTML документ представляется в виде дерева узлов, которыми являются элементы, атрибуты и текст.



Узлы дерева HTML документа

Согласно модели DOM:

- Весь документ представляется узлом документа;
- Каждый HTML тэг является узлом элемента;
- Текст внутри HTML элементов представляется текстовыми узлами;
- Каждому HTML атрибуту соответствует узел атрибута;
- Комментарии являются узлами комментариев.

```

<html>
  <head>
    <title>HTML документ</title>
  </head>
  <body>
    <h1>Заголовок </h1>
    <p>Просто текст</p>
  </body>
</html>
  
```

В этом примере корневым узлом является тэг `<html>`. Все остальные узлы содержатся внутри `<html>`. У этого узла имеется два дочерних узла: `<head>` и `<body>`. Узел `<head>` содержит узел `<title>`, а узел `<body>` содержит узлы `<h1>` и `<p>`.

Следует обратить особое внимание на то, что текст, расположенный в узле элемента соответствует текстовому узлу. В примере `<title>HTML документ</title>` узел элемента `<title>` содержит текстовый узел "HTML документ", то есть "HTML документ" не является значением элемента `<title>`. Тем не менее, в рамках HTML DOM значение текстового узла может быть доступно посредством свойства `innerHTML`.

Все узлы HTML документа могут быть доступны посредством дерева, при этом их содержимое может быть изменено или удалено, а также можно добавить новые элементы.

Все узлы дерева находятся в *иерархических отношениях* между собой. Для описания этих отношений используются термины *родитель*, *дочерний элемент* и *потомок*. Родительские узлы имеют дочерние узлы, а дочерние элементы одного уровня называются потомками (братьями или сестрами).

В отношении узлов дерева соблюдаются следующие принципы:

- Самый верхний узел дерева называется корневым;
- Каждый узел, за исключением корневого, имеет ровно один родительский узел;
- Узел может иметь любое число дочерних узлов;
- Конечный узел дерева не имеет дочерних узлов;

- Потомки имеют общего родителя.

Программный интерфейс HTML DOM

В рамках DOM модели HTML можно рассматривать как множество узловых *объектов*. Доступ к ним осуществляется с помощью *JavaScript* или других языков программирования. Программный интерфейс DOM включает в себя набор стандартных *свойств* и *методов*.

Свойства представляют некоторые сущности (например, <h1>), а *методы* - действия над ними (например, добавить <a>).

К типичным свойствам DOM относятся следующие:

- ☐ `x.innerHTML` – внутреннее текстовое значение HTML элемента `x` ;
- ☐ `x.nodeName` – имя `x` ;
- ☐ `x.nodeValue` – значение `x` ;
- ☐ `x.parentNode` – родительский узел для `x` ;
- ☐ `x.childNodes` – дочерний узел для `x` ;
- ☐ `x.attributes` – узлы атрибутов `x`.

Узловой объект, соответствующий HTML элементу поддерживает следующие методы:

- ☐ `x.getElementById(id)` – получить элемент с указанным `id` ;
- ☐ `x.getElementsByTagName(name)` – получить все элементы с указанным именем тэга (`name`) ;
- ☐ `x.appendChild(node)` – вставить дочерний узел для `x` ;
- ☐ `x.removeChild(node)` – удалить дочерний узел для `x`.

Пример 3.

Для получения текста из элемента <p> со значением атрибута `id "demo"` в HTML документе можно использовать следующий код:

```
txt = document.getElementById("demo").innerHTML
```

Тот же самый результат может быть получен по-другому:

```
txt=document.getElementById("demo").childNodes[0].nodeValue
```

В рамках DOM возможны 3 способа доступа к узлам:

1. С помощью метода `getElementById(ID)`. При этом возвращается элемент с указанным `ID`.
2. С помощью метода `getElementsByTagName(name)`. При этом возвращаются все узлы с указанным именем тэга (в виде индексированного списка). Первый элемент в списке имеет нулевой индекс.
3. Путем перемещения по дереву с использованием отношений между узлами.

Для определения длины списка узлов используется свойство `length`.

```
x = document.getElementsByTagName("p");
for (i = 0; i < x.length; i++)
{
    document.write(x[i].innerHTML);
    document.write("<br/>");
}
```

В данном примере внутрь HTML документа вставляется в виде списка текстовое содержимое всех элементов соответствующих тэгу <p>.

Для навигации по дереву в ближайших окрестностях текущего узла можно использовать следующие свойства:

- ☐ `parentNode` ;
- ☐ `firstChild` ;
- ☐ `lastChild`.

Для непосредственного доступа к тэгам можно использовать 2 специальных свойства:

- ❑ `document.documentElement` – для доступа к корневому узлу документа;
- ❑ `document.body` – для доступа к тэгу `<body>`.

Свойства узлов

В HTML DOM каждый узел является объектом, который может иметь методы (функции) и свойства. Наиболее важными являются следующие свойства:

- ❑ `nodeName` ;
- ❑ `nodeValue` ;
- ❑ `nodeType`.

Свойство `nodeName` указывает на имя узла. Это свойство имеет следующие особенности:

1. Свойство `nodeName` предназначено только для чтения;
2. Свойство `nodeName` узла элемента точно соответствует имени тэга;
3. Свойство `nodeName` узла атрибута соответствует имени атрибута;
4. Свойство `nodeName` текстового узла всегда равно `#text`
5. Свойство `nodeName` узла документа всегда равно `#document`

Замечание: `nodeName` всегда содержит имя тэга HTML элемента в верхнем регистре.

Свойство `nodeValue` указывает на значение узла. Это свойство имеет следующие особенности:

- ❑ Свойство `nodeValue` узла элемента не определено;
- ❑ Свойство `nodeValue` текстового узла указывает на сам текст;
- ❑ Свойство `nodeValue` узла атрибута указывает на значение атрибута.

Свойство `nodeType` возвращает тип узла. Это свойство предназначено только для чтения:

Наиболее важными типами узлов являются следующие:

Тип элемента	Тип узла
Element	1
Attribute	2
Text	3
Comment	8
Document	9

Изменение HTML элементов

HTML элементы могут быть изменены с посредством использования JavaScript, HTML DOM и событий.

В примере показано, как можно динамически изменять текстовое содержимое тэга `<p>`:

```
<html>
  <body>
    <p id="p1">Hello World!</p>
    <script type="text/javascript">
      document.getElementById("p1").innerHTML="New text!";
    </script>
  </body>
</html>
```

Диалоговые элементы

В JavaScript поддерживается работа со следующими диалоговыми элементами интерфейса:

1. **Alert.** Применяется для уведомления пользователя, работающего с веб-браузером.

Синтаксис:

```
alert("сообщение");
```

2. **Confirm.** Применяется для выбора пользователем одного из двух вариантов ответа "Да/Нет". Соответственно Confirm возвращает значение true/false.

Синтаксис:

```
confirm("вопрос");
```

3. **Prompt.** Применяется для ввода пользователем значения. При нажатии "ОК" возвращается введенное значение, в случае "Cancel" возвращается значение null.

Синтаксис:

```
prompt("вопрос/запрос","значение по умолчанию");
```

Ниже приводится код веб-страницы, в которой *пользователь* имеет возможность выбрать *цвет текста* с помощью диалогового элемента

```
<html>
<body>
// здесь будет отображаться текст
<div id="c" style="color:blue">Вы выбрали цвет текста: черный</div>

<script language="JavaScript">
// пользователь выбирает цвет текста
var tcolor = prompt("Выберите цвет текста: red, blue, green, yellow,
black", "black");
// задается текст
document.getElementById("c").innerHTML = "Вы выбрали цвет текста: " +
tcolor;
// задается цвет текста
document.getElementById("c").style.color = tcolor;

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

ХОД РАБОТЫ

1. Подготовьте на основе описанных выше примеров соответствующие веб-страницы и просмотрите их с помощью веб-браузера.

2. Контрольное задание. Создание таблицы случайно выбранных цветов

Взяв за основу *сценарий* построения таблицы умножения, постройте таблицу случайно выбранных цветов. Цвет ячейки таблицы задается с помощью атрибута **bgcolor**. Цвет ячейки описывается в рамках трехкомпонентной модели *RGB*, например: `<td bgcolor="#c0a145">`. Для генерации каждой компоненты можно использовать *генератор* случайных чисел с помощью методов объекта **Math** и преобразование в шестнадцатиричный формат:

```
color = Math.round(255.0*Math.random());
```

```
r = color.toString(16);
```

Результирующий цвет образуется путем конкатенации компонентов:

```
color = r + g + b;
```

Примерный вид результата работы сценария:

6852a0	1e6eac	2fala8	a378b5	34b1e0	238a8b	378c56	619f8d	581d8
11392a	59d966	ba33aa	631033	a33c65	636319	2fae43	4611f8	7f7794
ba7a67	cacb60	6d7160	3b1cb3	265979	b6bc2e	ff26ce	2d59d	6e4e1
7b677a	c4a6bc	bec14f	85437d	1a100f	583c37	4ea14	852213	59f120
909eb6	f395f	3b130	6c083	33310	41a531	d93bf1	1a3ec9	aab213
325f4	8fed	3693	d054dc	f2c484	ac8ef	3aea6	80afbe	e4bcd6
b03872	b13fe	5d7966	71585a	9c845e	233be1	8fded	ab4870	18ccf4
2e4287	cdf15f	927c81	8b7d6e	33c468	eec091	feba0	7b739c	2df9df
90a82d	a66dcf	a7a5f0	5a77f7	b1e64a	9658c7	d5cb45	ea2e82	b1b02d

Часть 2 Клиентские сценарии. Использование регулярных выражений

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Обработка событий в JavaScript

Популярность JavaScript во многом обусловлена именно тем, что написанный на нем *сценарий* может реагировать на действия пользователя и другие внешние события. Каждое из событий связано с тем или иным объектом: формой, гипертекстовой ссылкой или даже с окном, содержащим текущий документ.

В качестве примеров внешних событий, на которые могут *реагировать* объекты JavaScript, можно привести следующие.

- окончание загрузки документа в окно (или окончание загрузки документов во все фреймы окна). Это событие связано с объектом window;
- щелчок мышью на объекте. Это событие может быть связано с интерактивным элементом формы или с гипертекстовой ссылкой;
- получение объектом фокуса ввода. Это событие может быть связано с объектами типа Text, Password и с другими интерактивными элементами;
- передача на сервер данных, введенных пользователем с помощью интерактивных элементов. Связывается с формой.

Обработка события производится с помощью специально предназначенного для этого фрагмента кода, называемого обработчиком события. Для каждого события JavaScript предоставляет свой обработчик. Однако при построении сценария можно создавать собственный обработчик события и использовать его вместо обработчика, заданного по умолчанию.

Имя обработчика определяет, какое событие он должен обрабатывать. Так, для того чтобы *сценарий* нужным образом отреагировал на щелчок мышью, используется обработчик с именем **onClick**, для обработки события, заключающегося в получении фокуса ввода, - обработчик **onFocus**.

Для того чтобы указать интерпретатору JavaScript на то, что обработкой события должен заниматься обработчик, необходимо включить в *HTML-дескриптор* следующее *выражение*:

имя_обработчика="команды_обработчика"

Это *выражение* включается в тэг, описывающий *объект*, с которым связано событие.

Например, если необходимо обработать событие, заключающееся в получении фокуса полем ввода, *дескриптор*, описывающий этот интерактивный элемент, должен иметь примерно следующий вид:

```
<input type="text" name="Inform" onFocus="handleFocus();">
```

Имя обработчика является одним из атрибутов *HTML*-дескриптора, а команды, предназначенные для обработки события, выступают в роли значения этого атрибута. В данном случае обработка события производится в теле функции `handleFocus()`. В принципе, обработчиком может быть не только *функция*, но и любая последовательность команд JavaScript в виде составного оператора.

Следующий пример демонстрирует обработку события, связанного с наведением курсора мыши на гиперссылку:

```
<a href = "http://www.myhp.edu"
  onmouseover="alert('An onMouseOver event'); return false">
  
</a>
```

Ниже приводится полный текст *HTML* документа с JavaScript сценарием, в котором обрабатывается событие нажатия кнопки мыши, и определяется, какая именно из них была нажата:

```
<html>
<head>
  <script language = "javascript">
    function whichButton(event)
    {
      if (event.button == 2)
      {
        alert("Вы щелкнули правой кнопкой мыши!");
      }
      else
      {
        alert("Вы щелкнули левой кнопкой мыши!");
      }
    }
  </script>
</head>

<body onmousedown="whichButton(event)">
  <p>Щелкните любой кнопкой мыши в любом месте документа</p>
</body>
</html>
```

Таким образом, для того чтобы обработать какое-либо стандартное событие в браузере, необходимо добавить в подходящий *HTML* тег *атрибут*, соответствующий этому событию, указав в качестве значения атрибута имя JavaScript функции.

Атрибут	Описание
<code>onabort</code>	Прерванная загрузка изображения
<code>onblur</code>	утрата фокуса элементом
<code>onchange</code>	Изменение содержимого в поле ввода
<code>onclick</code>	Щелчок мыши на объекте
<code>ondblclick</code>	Двойной щелчок мыши на объекте
<code>onerror</code>	Ошибка при загрузке изображения или документа
<code>onfocus</code>	Получение фокуса элементом
<code>onkeydown</code>	Нажатие клавиши
<code>onkeypress</code>	Клавиша нажата

onkeyup	Отжатие клавиши
onload	Завершение загрузки страницы или изображения
onmousedown	Нажатие кнопки мыши
onmousemove	Перемещение курсора мыши
onmouseout	Смещение курсора мыши с объекта
onmouseover	Наведение курсора мыши на объект
onmouseup	Отжатие кнопки мыши
onreset	Кнопка "Reset" нажата
onresize	Изменение размера окна
onselect	Выделение текста
onsubmit	Кнопка "Submit" нажата
onunload	Уход с веб-страницы

Регулярные выражения

Регулярные выражения — система поиска текстовых фрагментов в электронных документах, основанная на специальной системе записи образцов для поиска.

Образец, задающий правило поиска, называется "шаблоном". Применение регулярных выражений принципиально преобразило технологии электронной обработки текстов.

С помощью регулярных выражений можно задавать структуру искомого шаблона и его позицию внутри строки (например, в начале или в конце строки, на границе или не на границе слова).

При описании структуры шаблона используются:

- ▣ гибкая система *квантификаторов* (операторов повторения);
- операторы описания наборов символов и их типа (числовые, нечисловые, специальные).

Для того, чтобы задать положение искомого фрагмента внутри строки, можно использовать один из следующих операторов:

Представление	Позиция
^	Начало строки
\$	Конец строки
\b	Граница слова
\B	Не граница слова
(?=шаблон)	Искомая строка следует после указанной строкой (с просмотром вперед)
(?!=шаблон)	Искомая строка не следует после указанной строки (с просмотром вперед)
(?<=шаблон)	Искомая строка следует после указанной строкой (с просмотром назад)
(?<!шаблон)	Искомая строка не следует после указанной строки (с просмотром назад)

Кроме того, язык регулярных выражений предоставляет набор *квантификаторов*, позволяющих указать число повторений шаблона:

Представление	Число повторений
---------------	------------------

{ n }	Ровно n
{ m, n }	От m до n включительно
{ $m,$ }	Не менее m
{ $, n$ }	Не более n

Имеются и более простые *квантификаторы*:

Представление	Число повторений	Эквивалент
*	Ноль или более	{ 0, }
+	Одно или более	{ 1, }
?	Ноль или одно	{ 0, 1 }

Для задания внутри шаблона группы символов можно использовать следующие *операторы*:

Оператор	Описание
[xyz]	Любой символ из указанного множества
[^xyz]	Любой символ не входящий в указанное множество
[x-z]	Любой символ из указанного диапазона
[^x-z]	Любой символ не входящий в указанный диапазон
. (точка)	Любой символ кроме символов разрыва или переноса строки
\w	Любой буквенно-цифровой символ, включая символ подчеркивания
\W	Любой не буквенный символ
\d	Любая цифра
\D	Любой нецифровой символ
\s	Любой неотображаемый символ
\S	Любой символ (кроме неотображаемых символов)

Для группировки отдельных частей шаблона можно использовать следующие *операторы*:

Оператор	Описание
()	Поиск группы символов внутри скобок и сохранение найденного соответствия
(?:)	Поиск группы символов внутри скобок без сохранения найденного соответствия
	Комбинирование частей в одно выражения с последующим поиском любой из частей в отдельности. Аналогично оператору "ИЛИ".

Если *шаблон* поиска включает специальные (как правило неотображаемые) символы, для их описания можно использовать следующие обозначения:

Обозначение	Описание
\0	Символ с нулевым кодом
\n	Символ новой строки
\r	Символ начала строки
\t	Символ табуляции
\v	Символ вертикальной табуляции
\xxx	Символ, имеющий заданный восьмеричный ASCII код <i>xxx</i>
\xdd	Символ, имеющий заданный шестнадцатиричный ASCII код <i>dd</i>
\uxxxx	Символ, имеющий ASCII код выраженный ЮНИКОДОМ <i>xxxx</i>

Квантификаторам в регулярных выражениях соответствует максимально длинная строка из возможных (т.е. *квантификаторы* являются "жадными"). Это может приводить к некоторым проблемам. Например, *шаблон* (<.*>) описывающий на первый взгляд теги *HTML* на самом деле будет выделять более крупные фрагменты в документе.

Например, строка вида

```
<p>
  <font color='blue'>
    <i>Регулярные выражения<i>
  </font>
  - удобный инструмент для поиска в строках
</p>
```

формально соответствует указанному выше шаблону

Для решения данной проблемы можно использовать два подхода.

1. В регулярном выражении учитываются символы, не соответствующие желаемому образцу (например, <[^>]*> для вышеописанного случая).

2. Определение квантификатора как *нежадного* (ленивого) - большинство реализаций позволяют это сделать, добавив после него знак вопроса.

Например, по шаблону (<.*?>) будут найдены все теги из рассмотренной строки.

Таким образом, получаются следующие "нежадные" модификации квантификаторов:

Модификатор	Описание
*?	"не жадный" эквивалент *
+?	"не жадный" эквивалент +
{n,}?	"не жадный" эквивалент {n,}

Следует, однако, иметь в виду, что использование "ленивых" *квантификаторов* может привести к ситуации, когда выражению соответствует слишком короткая, в частности, пустая строка.

Использование регулярных выражений в JavaScript

При поиске по тексту можно использовать *шаблон*, описывающий подстроку. В JavaScript такой *шаблон* может быть описан с помощью объекта *RegExp*. В простейшем случае такой *шаблон* описывает отдельный символ, однако имеет смысл его использовать для регулярных выражений.

Следующий ниже код описывает *RegExp* объект с именем *pttn*, содержащий регулярное *выражение*, описывающее целое десятичное число:

```
var pattn = new RegExp("/[0-9]+/");
```

Объект *RegExp* имеет 3 встроенных метода: *test()*, *exec()* и *compile()*.

☐ Метод *test()* выполняет поиск по шаблону:

```
var pattn = new RegExp("[0-9]+");
document.write(pattn.test("38 попугаев"));
```

Результат:

```
true
```

☐ Метод *exec()* выполняет поиск подстроки по шаблону и возвращает найденные соответствия; если соответствий нет, возвращается значение *null*:

```
var pattn=new RegExp("[0-9]+");
document.write(pattn.exec("38 попугаев"));
```

Результат:

```
38
```

Если необходимо найти все соответствия, то при вызове конструктора *RegExp* следует указать дополнительный параметр *"g"*, указывающий на необходимость глобального поиска:

```
var pattn = new RegExp("[0-9]+", "g");
do
{
result = pattn.exec("1 попугай, 2 попугая,..., 38 попугаев");
document.write(" " + result);
}
while (result != null)
```

Пример 7.2.

Результат:

```
1 2 38 null
```

☐ Метод *compile()* применяется для изменения ранее созданного шаблона:

```
var pattn = new RegExp("[0-5]+");
document.write(pattn.exec("38 попугаев"));
pattn.compile("[6-9]+");
document.write(";" + pattn.exec("38 попугаев"));
```

Пример 7.3.

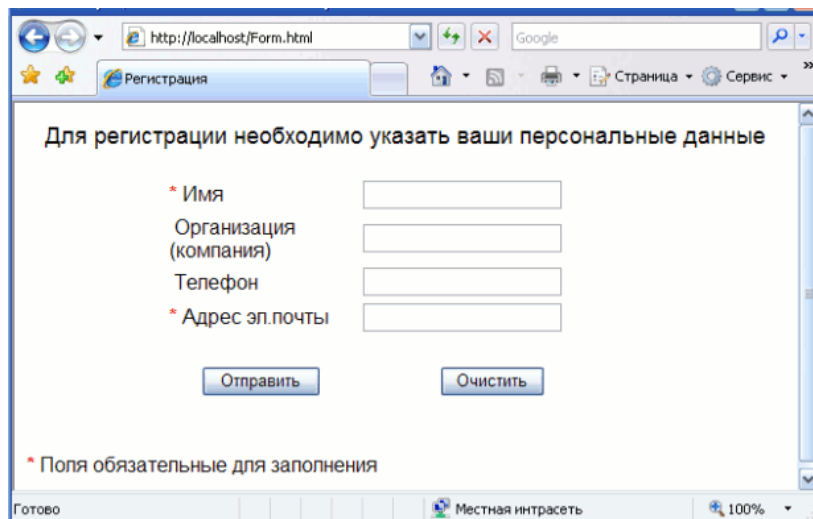
Результат:

```
3;8
```

ХОД РАБОТЫ

Проверка значений, введенных пользователем в поля формы для регистрации.

- Для выполнения работы необходимо создать веб-страницу, содержащую форму с полями, следующего вида:



- В тэге `<form>` добавьте обработчик события отправки данных вида:
`onSubmit = "CheckData(); return false;"`

В данном случае указана функция обработчик `CheckData()`. Оператор `return false;` предотвращает автоматическую отставку данных после выполнения функции-обработчика. Отправка данных будет выполняться из обработчика.

- Добавьте на странице секцию JavaScript кода, описывающего функцию-обработчик:

```
function CheckData()
{
    var ans;
    ans = confirm("Вы уверены, что хотите отправить введенные
данные ?");
    if (ans) submit();
}
```

Как это видно из кода, функция `CheckData()` в случае подтверждения со стороны пользователя самостоятельно вызывает метод `submit()` для передачи данных из формы.

- Теперь необходимо добавить проверку значений, введенных в поля формы пользователем.

Прежде всего, необходимо убедиться в том, что заполнены все поля, обязательные для ввода. Для этого можно использовать проверку на равенство нулю длины строки (свойство `length`), являющейся значением узла дерева документа, соответствующего полю ввода, например:

```
document.getElementById("uname").value.length.
```

Следующая проверка должна контролировать структуру и содержимое полей. Для этого можно использовать объект `RegExp`, например:

```
var validEMail, pattn;
```

```
pattn = new RegExp("^[\\._-A-Za-z0-9]+?@[\\._-A-Za-z0-9]+?\\.?[A-Za-z0-9]{2,6}$");
```

```
validEMail = pattn.test(document.getElementById("email").value));
```

В данном фрагменте описана проверка структуры электронного адреса из поля формы с идентификатором "email". Для проверки был использован шаблон на основе регулярного выражения.

Задание: Самостоятельно постройте регулярное *выражение*, описывающее *шаблон* для проверки номера телефона, и внесите все необходимые изменения и дополнения в функцию `CheckData()`.

Практическое занятие №25 Контроль и оценка знаний

Цель занятия: оценка сформированных знаний, умений и навыков.

1 Необходимо предоставить и продемонстрировать индивидуальный сайт, созданный с применением HTML, CSS.

2 Выполнить практическое задание по разработке сценария на языке JavaScript.

Приложение А

Статья из книги "Физики смеются. Но смеются не только физики", М.: Совпадение, 2020. ФИЗИКИ СМЕЮТСЯ

Эффект Чизхолма

Основные закономерности срывов, неудач и затяжек

Ф. Чизхолм

Можно быть уверенным только в одном: ни в чем нельзя быть уверенным. Если это утверждение истинно, оно тем самым и ложно.

Древний парадокс

Подобно большинству научных открытий, общие принципы, сформулированные в настоящей работе, покоятся на экспериментальных данных, в болезненном процессе накопления которых участвовало несколько поколений наблюдателей. Мой приятный долг поблагодарить их за объемистые записки, в которых зарегистрировано все, что касается разного рода проволочек и провалов. Это целая гора данных, и до сих пор не было строгой теории, которая связала бы их в цельную науку.

Я не хочу сказать, что ощущался недостаток в попытках объяснить, что именно происходит, когда люди стараются довести какое-то дело до конца. Уже в средние века фортуна считали капризной богиней, и Шекспир был близок к сути дела, когда называл ее "непостоянной". Строго научное объяснение рассматриваемого феномена стало возможным только в наше время. Разница между ожидаемыми и получаемыми результатами, как оказалось, может быть записана в виде точного соотношения, называемого уравнением Снэйфу и содержащего постоянную Финэйгла. Организация под названием "Международная организация инженеров-философов" уже опубликовала некоторые свои наблюдения: *"Какой бы расчет вы ни делали, любая ошибка, которая может в него вкратиться, - вкрадется"* и *"Любое устройство, требующее наладки и регулировки, с максимальным трудом поддается и тому, и другому"*.

Остается только обобщить эти и многие другие наблюдения, сделанные в различных специальных областях, и записать стоящий за ними совершенно общий, всеобъемлющий принцип, которому подчиняется во всех случаях целенаправленная человеческая деятельность. Это обобщение я называю первым законом Чисхолма: ВСЕ, ЧТО МОЖЕТ ИСПОРТИТЬСЯ, - ПОРТИТСЯ.

Дальнейшие исследования показывают, что логика, которой подчиняются рассматриваемые нами явления, не Аристотелева, поскольку следствие первого закона Чисхолма имеет следующий вид: ВСЕ, ЧТО НЕ МОЖЕТ ИСПОРТИТЬСЯ, - ПОРТИТСЯ ТОЖЕ.

Все, кому приходится иметь дело с планами, проектами и программами, сразу заметят, какой порядок наводят эти простые утверждения в хаосе их собственных неудач. Действительно, эти обобщения отличаются той классической простотой, по которой мы сразу узнаем фундаментальные открытия типа

$$E=mc^2.$$

Администраторы, футбольные тренеры, генералы и жены, пытающиеся перевоспитать своих мужей, сразу вынуждены будут признать (каждый для своего поля деятельности) справедливость первого закона.

Давно известно, что в *физических системах энтропия* (мера беспорядка) стремится к увеличению, и что системы с большой *энтропией* теряют ее в борьбе с менее организованным окружением. Аналог этого второго закона термодинамики действует и в жизни. Достаточно вспомнить, как с течением времени нарастает беспорядок на письменном столе после новогодней уборки. Поэтому я формулирую в самом общем виде второй закон Чисхолма: **КОГДА ДЕЛА ИДУТ ХОРОШО, ЧТО-ТО ДОЛЖНО ИСПОРТИТЬСЯ В САМОМ БЛИЖАЙШЕМ БУДУЩЕМ.**

У этого закона также есть очевидное следствие: **КОГДА ДЕЛА ИДУТ ХУЖЕ НЕКУДА, В САМОМ БЛИЖАЙШЕМ БУДУЩЕМ ОНИ ПОЙДУТ ЕЩЕ ХУЖЕ.**

Без труда можно получить и второе следствие: **ЕСЛИ ВАМ КАЖЕТСЯ, ЧТО СИТУАЦИЯ УЛУЧШАЕТСЯ, ЗНАЧИТ, ВЫ ЧЕГО-ТО НЕ ЗАМЕТИЛИ.**

По традиции фундаментальные научные законы объединяются по три, поэтому я поспешу сформулировать третий закон Чисхолма. Предварительная работа в этой области проведена многими лекторами, писателями, председателями комиссий и влюбленными, которые часто замечают, что люди слышат от вас вещи, которые вы им не говорили. Итак, обобщая: **ЛЮБУЮ ЦЕЛЬ ЛЮДИ ПОНИМАЮТ ИНАЧЕ, ЧЕМ ЧЕЛОВЕК, ЕЕ УКАЗЫВАЮЩИЙ.**

Следствие первое: **ЕСЛИ ЯСНОСТЬ ВАШЕГО ОБЪЯСНЕНИЯ ИСКЛЮЧАЕТ ЛОЖНОЕ ТОЛКОВАНИЕ, ВСЕ РАВНО КТО-ТО ПОЙМЕТ ВАС НЕПРАВИЛЬНО.**

Следствие второе: **ЕСЛИ ВЫ УВЕРЕНЫ, ЧТО ВАШ ПОСТУПОК ВСТРЕТИТ ВСЕОБЩЕЕ ОДОБРЕНИЕ, КОМУ-ТО ОН НЕ ПОНРАВИТСЯ.**

Учет законов Чисхолма как решающих факторов при планировании любого процесса должен понизить всеобщее нервное напряжение и решить национальную проблему перепроизводства адреналина.

Из книги "A Stress Analysis of a Strapless Evening Gown"

Englewood Cliffs, 1963.

Приложение Б

Фрагмент из книги "Полное собрание законов Мерфи", Мн.: ООО "Попурри", 2005

Законы Мерфи. Книга 1. ДЕЛА КОНСТРУКТОРСКИЕ ЗАКОН ТЕХНИЧЕСКИХ УСЛОВИЙ КЛИНШТЕЙНА

В технических условиях Закон Мерфи отменяет закон Ома.

ЗАКОН УТЕРЯННОГО ДЮЙМА

При разработке любого типа конструкции в пятницу после 16.40 ни один габаритный размер нельзя подсчитать точно.

Следствия.

1. При тех же условиях, если малые размеры даны с точностью до шестнадцатой дюйма, даже приблизительное определение габаритного размера невозможно.
2. В понедельник в 9.01 точное значение станет очевидным само по себе.

ЗАКОНЫ ПРИКЛАДНОЙ ПУТАНИЦЫ

1. Деталь, которую завод забыл поставить, составляет 75% поставок.

Следствие.

Завод не только забыл ее поставить, но и не изготовлял ее в течение 50% рабочего времени.

2. Доставка, которая обычно занимает один день, длится пять, если вы ждете товара.
3. Если вы добавили к графику две недели на непредвиденные задержки, не забудьте еще две на непредвиденные непредвиденные задержки.
4. В любой конструкции на самой важной детали, скорее всего, стоит неправильная метка.

Следствия.

- В любом наборе деталей с одинаковой меткой для сборки найдется деталь, на которой эта метка лишняя.
- Это не обнаруживается до тех пор, пока вы не попытаетесь вставить ее туда, куда указывает метка.
- Никогда не спорьте с заводом-производителем об ошибке. Они очень тщательно проверяют акты о приемке продукции.

ДЕЛА МАШИННЫЕ ЗАКОН МАСТЕРСКИХ ЭНТОНИ

Любой упавший инструмент закатывается в самый недоступный уголок мастерской.

Следствие.

По пути в этот уголок инструмент сперва непременно попадет вам по ноге.

ПЕРВЫЙ ЗАКОН ДЖОНСОНА

Любое механическое устройство отказывает в самый неподходящий момент.

ЗАКОН РАЗДРАЖЕНИЯ

Если во время работы вы убрали инструмент, который, вы уверены, больше не понадобится, в нем тут же возникает срочная потребность.

ЗАКОН САТТИНГЕРА

Если вставить вилку в розетку, прибор будет работать лучше.

ЗАКОН ШМИДТА

Любая вещь, если с ней долго возиться, обязательно сломается.

Артур Блох.

Приложение В

Статья из книги "Физики смеются. Но смеются не только физики", М.: Совпадение, 206.

Но смеются не только физики... Чем заняты физики

Шагая в ногу со временем, редакция стенгазеты "Импульс" Физического института АН СССР сформировала отдел социологических исследований. Сотрудники этого отдела провели опрос населения Москвы на тему "Чем заняты физики".

Группа населения	Опрошено	Отвечили	Не знают	Ответы
Писатели-реалисты	11	7	4	Спорят до хрипоты в прокуренных комнатах. Неизвестно зачем ставят непонятные и опасные опыты на огромных установках
Писатели-фантасты	58	58	0	Работают на громадных электронных машинах, именуемых электронным мозгом. Работают преимущественно в космосе
Студенты первого курса	65	65	0	Очень много размышляют. Делают открытия не реже раза в месяц
Студенты-дипломники	30	10	20	Паяют схемы
Младшие научные сотрудники-экспериментаторы	53	40	13	Просят старших найти течь. Пишут статьи. Бегают в отдел снабжения. Моют форвакуумные насосы. Хлопают ушами на семинарах.
Младшие научные сотрудники-теоретики	19	19	0	Разговаривают в коридорах, надеясь сделать великое открытие. Пишут множество формул, большая часть которых окажется неверной
Старшие научные сотрудники	7	6	1	Спят на заседаниях. Помогают младшим научным сотрудникам искать течь.
Сотрудники отдела кадров	5	5	0	Экспериментаторы должны приходиться в 8.25, чтобы в 8.30 уже молча сидеть возле включенных установок. Теоретики вовсе не работают, их на месте не застанешь
Сотрудники охраны	6	6	0	Ходят взад-вперед. Предъявляют пропуск вверх ногами
Сотрудники Министерства финансов	18	18	0	Тратят деньги впустую