МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ (ФИЛИАЛ)
ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО
ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
В Г. ТАГАНРОГЕ РОСТОВСКОЙ ОБЛАСТИ
ПИ (филиал) ДГТУ в г. Таганроге

ЦМК «Прикладная информатика»

Практикум

По выполнению практических работ

по профессиональному МДК:

МДК 08.01 Проектирование и разработка интерфейсов пользователя

для специальности 09.02.07 Информационные системы и программирование,

квалификации

«Разработчик веб и мультимедийных приложений»

Таганрог
2023

Составители: А.А. Погорелов

Практикум по выполнению практических работ по МДК:
МДК 08.01 Проектирование и разработка интерфейсов пользователя . ПИ (филиала) ДГТУ в г.Таганроге, 2023г.

В практикуме кратко изложены теоретические вопросы, необходимые для успешного выполнения практических работ, рабочее задание и контрольные вопросы для самопроверки.

Предназначено для обучающихся по специальности 09.02.07 «Информационные системы и программирование». Квалификации выпускника: «Разработчик веб и мультимедийных приложений»

Ответственный за выпуск:

Председатель ЦМК: _____ О.В. Андриян

Практическая работа № 1

«Составление технического задания на разработку web-сайта»

Количество часов на выполнение: 4, из них на практическую подготовку 4

Оборудование: ОС Windows 7, 10; Microsoft Word

Задание:

1 Цель работы: научиться правильно составлять техническое задание на разработку web-сайта

2 Пояснение к работе

2.1 Краткие теоретические сведения

Техническое задание – первый и самый важный шаг на пути создания сайта. Это документ на основе которого Исполнитель разрабатывает сайт, а Заказчик оценивает качество готового продукта.

Техническое задание является неотъемлемой частью договора на разработку сайта и не может быть подправлено в течение всего периода создания web-сайта.

Техническое задание разделяется на три части:

- Назначение и цели создания сайта.
- Содержание сайта.
- Структура сайта.

Образец технического задания на разработку web-сайта Техническое задание на разработку Веб-сайта

1. Имя сайта (название домена). www._____.ru Если домен www._____.ru будет занят, возможна замена имени.

2. Название сайта. Сайт ООО «_____». Далее - Фирма.

3. Назначение сайта (цель создания сайта). Представление Фирмы в Интернет: информация о Фирме, история Фирмы, партнёры Фирмы, Заказчики Фирмы, цены на оказываемые услуги, справочная техническая и юридическая информация, советы клиентам, сопроводительные графические рисунки, юридический адрес, почтовый адрес, схема проезда, контактная информация, банковские реквизиты, сведения об имеющихся вакансиях. Сайт должен способствовать привлечению клиентов и нахождению деловых партнеров.

4. Язык сайта. Русский

5. Основные ключевые слова, по которым сайт должны находить по запросам в поисковых системах и Интернет - каталогах. Согласно материалам Заказчика. Примечание: Перечень ключевых слов для Веб-дизайнера сайта носит справочный характер и не входит в число обязательных параметров, подлежащих проверке при приемке сайта. Занимаемые сайтом позиции в рейтингах, каталогах и поисковых системах не оговариваются.

6. Объём и состав текстовой и графической информации в электронном виде. Согласно материалам Заказчика.

7. Предполагаемая возрастная аудитория сайта. От 30 лет и старше.

7.1. Предполагаемое возрастное ядро аудитории от 35 до 50 лет.

7.2. Данная информация носит рекомендательный характер. Цифровые показатели контролю и проверке при приёмке сайта не подлежат.

8. Количество страниц сайта. Сайт должен содержать следующие html страницы:

1) Главная (домашняя) страница;
2) Прайс-лист;
3) Фото (каталог) товаров;
4) 4,5,6,7,8,9,10 - Справочная информация;
11) О Фирме;
12) Офис;
13) Партнёры;
14) Вакансии;

15) Потребности;
16) Сервисы. Количество html страниц сайта определяется Веб-дизайнером самостоятельно, исходя из объёма предоставленных материалов Заказчика.

9. Кнопки управления (навигация сайта). Определяются Веб-дизайнером самостоятельно. С каждой страницы сайта должен быть обеспечен переход (установлена гиперссылка) на главную страницу сайта. Сайт должен содержать страницу "Содержание" (карта сайта).
10. Блок схема сайта. Определяется Веб-дизайнером самостоятельно. Головная (начальная) страница сайта должна содержать гиперссылки, обеспечивающие переход с нее на не менее чем 95% страниц сайта, но не более чем 160 гиперссылок.
11. Объём сайта, Мб. Не оговаривается.
12. Оформление рисунков. Все рисунки объемом более 1 Кб должны быть выполнены с замещающим текстом. Рисунки размером более 15 Кб должны быть выполнены с предпросмотром. Формат всех рисунков gif или jpg (jpeg).
13. Пропускная способность линии связи. Среднее время загрузки страниц не должно превышать 28 секунд при скорости соединения 28.8 Кбит/сек. Допускается увеличение времени загрузки отдельных страниц до 36 секунд, но не более чем на 30% числа страниц сайта. Головная (начальная) страница должна иметь время загрузки не более 40 секунд. Примечание: Во всех случаях не учитывается время загрузки подгружаемых элементов (счетчики, баннеры, информеры и т.д.).
14. Основной диапазон разрешения мониторов, на которых будет просматриваться сайт. От 600x800 до 1240x1024 пикселей. Основное разрешение, на которое оптимизируется сайт: 1024x768 пикселей.
15. Минимальное разрешение монитора, на котором будет просматриваться сайт. 600 x 800 пикселей. При указанном разрешении возможность просмотра страниц сайта без горизонтальной прокрутки браузера не предусматривается.
16. Основной браузер, которым будет просматриваться сайт, и его минимальная версия. IE 5.5 и выше.
17. Цветовая палитра. Основной режим мониторов, на которых будет просматриваться сайт: 15 разрядов цветов и выше (число цветов 65536 и выше). При разработке сайта должен быть обеспечена возможность его просмотра при использовании безопасной цветовой палитры (разрядность цветов 8). Изменения оттенков цветов, при просмотре сайта с использованием безопасной цветовой палитры, не оговариваются.
18. Общий фон сайта. Общий фон сайта светлый (белый). Допускается использование светлого фонового рисунка.
19. Размер и вид шрифта сайта. Размер шрифта сайта должен быть в пределах 10-12 для оформления текста. Размер шрифта для оформления заголовков, названия страниц и т.д. не оговаривается. Вид (название) шрифта не оговаривается.
20. Регистрация сайта в каталогах, рейтингах, топах и пр. Оговаривается дополнительно.
21. Проведение рекламной кампании по раскрутке сайта. Раскрутка сайта определяется отдельным техническим заданием. В настоящем техническом задании раскрутка сайта не оговаривается и не входит в состав выполняемых работ (услуг).
22. Срок разработки сайта. Три недели со дня зачисления 70% предоплаты на расчётный счёт Веб-студии.
23. Порядок передачи сайта. Веб-дизайнер передает сайт на CD ROM, а также логин, пароль и название (код передачи данных) по протоколу ftp. Заказчик обязан проверить наличие грамматических и орфографических ошибок на сайте в течение трех рабочих дней. Обнаруженные ошибки Веб-дизайнер обязан устранить в течение трех рабочих дней.
24. Сопровождение сайта. Сопровождение сайта определяется отдельным техническим заданием. В настоящем техническом задании сопровождение сайта не оговаривается и не входит в состав выполняемых работ (услуг).
25. Дополнительные условия. Каждая страница сайта должна содержать логотип и

название Фирмы. Внизу на каждой странице сайта должна быть указана контактная информация. Сайт должен содержать не менее двух счетчиков подсчета посетителей.
Материалы предоставляемые Заказчиком:
Текстовая (формат Word) и графическая информация (формат jpeg и gif), представленные на CD ROM.
Примечание:
Задание на сайт может быть изменено с учетом конкретных требований.
Задание на сайт предназначено для русскоязычных сайтов, объемом не более 50 html страниц. Если сайт имеет версию на иностранном языке или версию для просмотра на мобильных устройствах, задание на сайт должно быть дополнено соответствующими пунктами.
Веб-дизайнер не несет ответственности за несоответствие сайта эстетическим ожиданиям Заказчика при условии выполнения технического задания на сайт.

Подписи Сторон:

Веб-студия: Заказчик:

2.2 Перечень используемого оборудования
2.2.1 Персональный компьютер
2.2.2 Описание практической работы
3 Задание
3.1 Составьте техническое задание на разработку web-сайта по вариантам:

№ варианта	Название сайта (тематика)
1	Личный сайт
2	Сайт туристической фирмы
3	Сайт Интернет магазина
4	Сайт библиотеки
5	Сайт новостей
6	Сайт погоды
7	Сайт спортивного клуба
8	Сайт выставочного комплекса
9	Сайт концертного зала
10	Сайт инструментального ансамбля
11	Сайт железнодорожного вокзала
12	Сайт телеграфа
13	Сайт фотоателье
14	Сайт транспортной компании
15	Сайт поликлиники
16	Сайт института
17	Сайт центрального рынка
18	Сайт ресторана
19	Сайт парка культуры и отдыха
20	Сайт журнала
21	Сайт коммерческой фирмы
22	Сайт Интернет аукциона
23	Сайт архива
24	Сайт трамвайного маршрута
25	Сайт web-студии
26	Сайт кадрового агентства
27	Сайт кафе

28	Сайт газеты
29	Сайт молодежной одежды
30	Сайт театра

4 Контрольные вопросы
4.1 Что такое техническое задание?
4.2 Как выглядит в общем виде техническое задание на разработку web-сайта
Требования к оформлению отчетного материала:
5 Содержание отчета
5.1 Наименование работы
5.2 Цель работы
5.3 Задание
5.4 Выводы по работе
5.5 Ответы на контрольные вопросы
Требования к оформлению отчетного материала: отчет о выполненной работе предоставлен с кодами и скриншотами веб-страниц; отчет содержит штамп
Форма контроля: отчет
Ссылки на источники: [2,3]

Практическая работа № 2
«Применение тегов HTML при создании web-страниц»

Количество часов на выполнение:6, из них на практическую подготовку 6
Оборудование: Windows, Notepade++, Chrome, Mozilla FireFox
Задание:
1 Цель работы:
1.1 Научиться определять основные теги HTML-документа, их атрибуты.
1.2 Определять их вид.
2 Пояснение к работе
2.1 Краткие теоретические сведения
Для освоения HTML Вам понадобятся две вещи:
1. Любой браузер, т.е., программа, пригодная для просмотра HTML-файлов. Например,Chrome или Mozilla FireFox.
2. Любой редактор текстовых файлов, поддерживающий русский язык в выбранной Вами кодировке. Если на Вашем компьютере установлен Windows, вполне подойдет Notepad++.
HTML-документ — это просто текстовый файл с расширением *.html. Вот самый простой HTML-документ:

```
<html>
<head>
<title>
Пример 1
</title>
</head>
<body>
<H1>
Привет!
</H1>
<P>
Это простейший пример HTML-документа.
</P>
<P>
Этот *.html-файл может быть одновременно открыт и в Notepad, и в Chrome или Mozilla FireFox .
```

Сохранив изменения в Notepad, просто нажмите кнопку F5 в Chrome или Mozilla FireFox , чтобы увидеть эти изменения реализованными в HTML-документе.
</P>
</body>
</html>
Как видно из примера, вся информация о форматировании документа сосредоточена в его фрагментах, заключенных между знаками «<» и ">". Такой фрагмент (например, <html>) называется меткой (по-английски — tag, читается "тэг").
Большинство HTML-меток — парные, то есть на каждую открывающую метку вида <tag> есть закрывающая метка вида </tag> с тем же именем, но с добавлением «/».
Многие метки, помимо имени, могут содержать атрибуты — элементы, дающие дополнительную информацию о том, как браузер должен обработать текущую метку. В нашем простейшем документе, однако, нет ни одного атрибута. Но мы обязательно встретимся с атрибутами уже в следующем разделе.
Обязательные метки <html> ... </html>
Метка <html> должна открывать HTML-документ. Аналогично, метка </html> должна завершать HTML-документ.
<head> ... </head>
Эта пара меток указывает на начало и конец заголовка документа. Помимо наименования документа (см. описание метки <title> ниже), в этот раздел может включаться множество служебной информации, о которой мы обязательно поговорим чуть позже.
<title> ... </title>
Все, что находится между метками <title> и </title>, толкуется браузером как название документа. Chrome или Mozilla FireFox , например, показывает название текущего документа в заголовке окна и печатает его в левом верхнем углу каждой страницы при выводе на принтер. Рекомендуется название не длиннее 64 символов.
<body> ... </body>
Эта пара меток указывает на начало и конец тела HTML-документа, каковое тело, собственно, и определяет содержание документа.
<H1> ... </H1> — <H6> ... </H6>
Метки вида <Hi> (где i — цифра от 1 до 6) описывают заголовки шести различных уровней. Заголовок первого уровня — самый крупный, шестого уровня, естественно — самый мелкий.
<P> ... </P>
Такая пара меток описывает абзац. Все, что заключено между <P> и </P>, воспринимается как один абзац.
Метки <Hi> и <P> могут содержать дополнительный атрибут ALIGN, например:
<H1 ALIGN=CENTER>Выравнивание заголовка
по центру</H1>
или
<P ALIGN=RIGHT>Образец абзаца с выравниванием
по правому краю</P>
Подытожим
<html>
<head>
<title>Пример 2</title>
</head>
<body>
<H1 ALIGN=CENTER>Привет!</H1>
<H2>Это чуть более сложный пример
HTML-документа</H2>
<P>Теперь мы знаем, что абзац можно выравнивать

не только влево, </P>
<P ALIGN=CENTER>но и по центру</P>
<P ALIGN=RIGHT>или по правому краю.</P>
</body>
</html>
С этого момента Вы знаете достаточно, чтобы создавать простые HTML-документы самостоятельно от начала до конца. В следующем разделе мы поговорим, как можно улучшить наш простой HTML-документ. Начнем с малого — с абзаца.
Непарные метки
В этом разделе мы поговорим о метках, которые не подчиняются двум основным правилам HTML: все они непарные, а некоторые (так называемые &-последовательности) к тому же должны вводиться только маленькими буквами.

Эта метка используется, если необходимо перейти на новую строку, не прерывая абзаца. Очень удобно при публикации стихов
<html>
<head>
<title>Пример 3</title>
</head>
<body>
<H1>Стих</H1>
<H2>Автор неизвестен</H2>
<P>Однажды в студеную зимнюю пору
Сижу за решеткой в темнице сырой.
Гляжу - поднимается медленно в гору
Вскормленный в неволе орел молодой.</P>
<P>И шествуя важно, в спокойствии чинном,
Мой грустный товарищ, махая крылом,
В больших сапогах, в полушубке овчинном,
Кровавую пищу клюет под окном.</P>
</body> </html>
<HR>
Метка <HR> описывает вот такую горизонтальную линию:

Метка может дополнительно включать атрибуты SIZE (определяет толщину линии в пикселах) и/или WIDTH (определяет размах линии в процентах от ширины экрана
<html>
<head>
<title>Пример 4</title>
</head>
<body>
<H1>Коллекция горизонтальных линий</H1>
<HR SIZE=2 WIDTH=100%>
<HR SIZE=4 WIDTH=50%>
<HR SIZE=8 WIDTH=25%>
<HR SIZE=16 WIDTH=12%>
</body>
</html>
&-последовательности
Поскольку символы "<" и ">" воспринимаются браузерами как начало и конец HTML-меток, возникает вопрос: а как показать эти символы на экране? В HTML это делается с помощью &-последовательностей (их еще называют символьными объектами или эскейп-

последовательностями). Браузер показывает на экране символ "<", когда встречает в тексте последовательность < (по первым буквам английских слов less than — меньше, чем). Знак ">" кодируется последовательностью > (по первым буквам английских слов greater than — больше, чем).
Символ "&" (амперсанд) кодируется последовательностью &
Двойные кавычки (") кодируются последовательностью "
Помните: точка с запятой — обязательный элемент &-последовательности. Кроме того, все буквы, составляющие последовательность, должны быть в нижнем регистре (т.е., маленькие). Использование меток типа " или & не допускается.
Вообще говоря, &-последовательности определены для всех символов из второй половины ASCII-таблицы (куда, естественно, входят и русские буквы). Дело в том, что некоторые серверы не поддерживают восьмибитную передачу данных, и поэтому могут передавать символы с ASCII-кодами выше 127 только в виде &-последовательностей.
Комментарии
Браузеры игнорируют любой текст, помещенный между <!-- и -->. Это удобно для размещения комментариев.
<!-- Это комментарий -->
Форматирование шрифта
HTML допускает два подхода к шрифтовому выделению фрагментов текста. С одной стороны, можно прямо указать, что шрифт на некотором участке текста должен быть жирным или наклонным, то есть изменить физический стиль текста. С другой стороны, можно пометить некоторый фрагмент текста как имеющий некоторый отличный от нормального логический стиль, оставив интерпретацию этого стиля браузеру. Поясним это на примерах.
Физические стили
Под физическом стилем принято понимать прямое указание браузеру на модификацию текущего шрифта. Например, все, что находится между метками <B> и </B>, будет написано жирным шрифтом. Текст между метками <I> и </I> будет написан наклонным шрифтом.
Несколько особняком стоит пара меток <TT> и </TT>. Текст, размещенный между этими метками, будет написан шрифтом, имитирующим пишущую машинку, то есть имеющим фиксированную ширину символа.
Логические стили
При использовании логических стилей автор документа не может знать заранее, что увидит на экране читатель. Разные браузеры толкуют одни и те же метки логических стилей по-разному. Некоторые браузеры игнорируют некоторые метки вообще и показывают нормальный текст вместо выделенного логическим стилем. Вот самые распространенные логические стили.
<EM> ... </EM>
От английского emphasis — акцент.
<STRONG> ... </STRONG>
От английского strong emphasis — сильный акцент.
<CODE> ... </CODE>
Рекомендуется использовать для фрагментов исходных текстов.
<SAMP> ... </SAMP>
От английского sample — образец. Рекомендуется использовать для демонстрации образцов сообщений, выводимых на экран программами.
<KBD> ... </KBD>
От английского keyboard — клавиатура. Рекомендуется использовать для указания того, что нужно ввести с клавиатуры.
<VAR> ... </VAR>
От английского variable — переменная. Рекомендуется использовать для написания имен

переменных.
Пример
Подытожим наши знания о логических и физических стилях.

```
<html>
<head>
<title>Пример 5</title>
</head>
<body>
<H1>Шрифтовое выделение фрагментов текста</H1>
<P>Теперь мы знаем, что фрагменты текста можно
выделять
<B>жирным</B> или <I>наклонным</I> шрифтом.
Кроме того, можно
включать в текст фрагменты с фиксированной шириной
символа
<TT>(имитация пишущей машинки)</TT></P>
<P>Кроме того, существует ряд логических стилей:</P>
<P><EM>EM - от английского emphasis - акцент </EM><BR>
<STRONG>STRONG - от английского strong emphasis -
сильный акцент </STRONG><BR>
<CODE>CODE - для фрагментов
исходных текстов</CODE><BR>
<SAMP>SAMP - от английского sample -
образец </SAMP><BR>
<KBD>KBD - от английского keyboard -
клавиатура</KBD><BR>
<VAR>VAR - от английского variable -
переменная </VAR></P>
</body>
</html>
```

Ненумерованные списки: <UL> ... </UL>
Текст, расположенный между метками <UL> и </UL>, воспринимается как ненумерованный список. Каждый новый элемент списка следует начинать с метки <LI>. Например, чтобы создать вот такой список:

- Пара 1
- Пара 2
- Пара 3

необходим вот такой HTML-текст:

```
<UL>
<LI>Пара 1;
<LI>Пара 2;
<LI>Пара 3
</UL>
```

Обратите внимание: у метки <LI> нет парной закрывающей метки.
Нумерованные списки: <OL> ... </OL>
Нумерованные списки устроены точно так же, как ненумерованные, только вместо символов, выделяющих новый элемент, используются цифры. Если слегка модифицировать наш предыдущий пример:

```
<OL>
<LI>Пара 1;
<LI>Пара 2;
<LI>Пара 3
```

</OL>
Списки определений: <DL> ... </DL>
Список определений несколько отличается от других видов списков. Вместо меток <LI> в списках определений используются метки <DT> (от английского definition term — определяемый термин) и <DD> (от английского definition definition — определение определения). Разберем это на примере. Допустим, у нас имеется следующий фрагмент HTML-текста:

```
<DL>
<DT>HTML
<DD>Термин HTML (HyperText Markup Language) означает
'язык маркировки гипертекстов'. Первую версию HTML
разработал сотрудник Европейской лаборатории физики
элементарных частиц Тим Бернерс-Ли.
<DT>HTML-документ
<DD>Текстовый файл с расширением *.html
</DL>
```

Обратите внимание: точно так же, как метки <LI>, метки <DT> и <DD> не имеют парных закрывающих меток.
Если определяемые термины достаточно коротки, можно использовать модифицированную открывающую метку <DL COMPACT>. Например, вот такой фрагмент HTML-текста:

```
<DL COMPACT>
<DT>А
<DD>Первая буква алфавита
<DT>Б
<DD>Вторая буква алфавита
<DT>В
<DD>Третья буква алфавита
</DL>
```

Вложенные списки
Элемент любого списка может содержать в себе целый список любого вида. Число уровней вложенности в принципе не ограничено, однако злоупотреблять вложенными списками все же не следует.
Вложенные списки очень удобны при подготовке разного рода планов и оглавлений.

```
<html>
<head>
<title>Пример 6</title>
</head>
<body>
<H1>HTML поддерживает несколько видов списков </H1>
<DL>
<DT>Ненумерованные списки
<DD>Элементы ненумерованного списка выделяются
специальным
символом и отступом слева:
<UL>
<LI>Элемент 1
<LI>Элемент 2
<LI>Элемент 3
</UL>
<DT>Нумерованные списки
<DD>Элементы нумерованного списка выделяются
```

отступом слева, а
также нумерацией:
<OL>
<LI>Элемент 1
<LI>Элемент 2
<LI>Элемент 3
</OL>
<DT>Списки определений
<DD>Этот вид списков чуть сложнее, чем два предыдущих,
но и выглядит более эффектно.
<P>Помните, что списки можно встраивать один в другой,
но не
следует закладывать слишком много уровней вложенности.
</P>
<P>Обратите внимание, что внутри элемента списка
может находиться
несколько абзацев. Все абзацы при этом будут иметь
одинаковое левое поле. </P>
</DL>
</body>
</html>

Форматированный текст: <PRE> ... </PRE>

Текст, заключенный между метками <PRE> и </PRE> (от английского preformatted — предварительно форматированный), выводится браузером на экран как есть — со всеми пробелами, символами табуляции и конца строки. Это очень удобно при создании простых таблиц.

Текст с отступом: <BLOCKQUOTE> ... </BLOCKQUOTE>

Текст, заключенный между метками <BLOCKQUOTE> и </BLOCKQUOTE>, выводится браузером на экран с увеличенным левым полем.

Связывание

Как уже упоминалось в самом начале, сокращение HTML означает "язык маркировки гипертекстов". Про маркировку мы уже поговорили достаточно. Не пора ли перейти к гипертексту?

Прежде всего, что же такое гипертекст? В отличие от обыкновенного текста, который можно читать только от начала к концу, гипертекст позволяет осуществлять мгновенный переход от одного фрагмента текста к другому. Системы помощи многих популярных программных продуктов устроены именно по гипертекстовому принципу. При нажатии левой кнопкой мыши на некоторый выделенный фрагмент текущего документа происходит переход к некоторому заранее назначенному документу или фрагменту документа.

В HTML переход от одного фрагмента текста к другому задается с помощью метки вида:

<A HREF="[адрес перехода]">
выделенный фрагмент текста</A>

В качестве параметра [адрес перехода] может использоваться несколько типов аргументов. Самое простое — это задать имя другого HTML-документа, к которому нужно перейти. Например:

<A HREF="index.html">Перейти к оглавлению</A>

Такой фрагмент HTML-текста приведет к появлению в документе выделенного фрагмента, при нажатии на который в текущее окно будет загружен документ index.html.

Обратите внимание: если в адресе перехода не указан каталог, переход будет выполнен внутри текущего каталога. Если в адресе перехода не указан сервер, переход будет выполнен на текущем сервере.

Из этого следует одно очень важное практическое соображение. Если Вы подготовили к публикации некоторую группу HTML-документов, которые ссылаются друг на друга только по имени файла и находятся в одном каталоге на Вашем компьютере, вся эта группа документов будет работать точно так же, если ее поместить в любой другой каталог на любом другом компьютере, на локальной сети или... на Интернет! Таким образом, у Вас появляется возможность разрабатывать целые коллекции документов без подключения к Интернет, и только после окончательной готовности, подтвержденной испытаниями, помещать коллекции документов на Интернет целиком.
При необходимости можно задать переход не просто к некоторому документу, но и к определенному месту внутри этого документа. Для этого необходимо создать в документе, к которому будет задан переход, некоторую опорную точку, или анкер. Разберем это на примере.
Допустим, что необходимо осуществить переход из файла 1.html к словам "Переход закончен" в файле 2.html (файлы находятся в одном каталоге). Прежде всего, необходимо создать вот такой анкер в файле 2.html:
<A NAME="AAA">Переход закончен</A>
Слова "Переход закончен" при этом никак не будут выделены в тексте документа.
Затем в файле 1.html (или в любом другом) можно определить переход на этот анкер:
<A HREF="2.html#AAA">Переход к анкеру AAA</A>
Кстати говоря, переход к этому анкеру можно определить и внутри самого документа 2.html — достаточно только включить в него вот такой фрагмент:
<A HREF="#AAA">Переход к анкеру AAA</A>
На практике это очень удобно при создании больших документов. В начале документа можно поместить оглавление, состоящее из ссылок на анкеры, расположенные в заголовках разделов документа.
Во избежание недоразумений рекомендуется задавать имена анкеров латинскими буквами. Следите за написанием имен анкеров: большинство браузеров отличают большие буквы от маленьких. Если имя анкера определено как AAA, ссылка на анкер aaa или AaA не выведет Вас на анкер AAA, хотя документ, скорее всего, будет загружен корректно.
Изображения в HTML-документе
Встроить изображение в HTML-документ очень просто. Для этого нужно только иметь это самое изображение в формате GIF (файл с расширением *.gif) или JPEG (файл с расширением *.jpg или *.jpeg) и одну строчку в HTML-тексте.
Допустим, нам нужно включить в документ изображение, записанное в файл picture.gif, находящийся в одном каталоге с HTML-документом. Тогда строчка будет вот такая:
<IMG SRC="picture.gif">
Метка <IMG SRC="[имя файла]"> может также включать атрибут ALT="[текст]", например:
<IMG SRC="picture.gif" ALT="Картинка">
Встретив такую метку, браузер покажет на экране текст Картинка и начнет загружать на его место картинку из файла picture.gif. Атрибут ALT может оказаться необходимым для старых браузеров, которые не поддерживают изображений, а также на случай, если у браузера отключена автоматическая загрузка изображений (при медленном подключении к Интернет это делается для экономии времени).
Файл, содержаций изображение, может находиться в другом каталоге или даже на другом сервере. В этом случае стоит указать его полное имя . Разберем все, что мы знаем об изображениях
<HTML>
<HEAD>
<TITLE>Пример 8</TITLE>
</HEAD>

```
<BODY>
<H1>Изображения </H1>
<P>Изображение можно встроить очень просто: </P>
<P><IMG SRC="picture.gif"></P>
<P>Кроме того, изображение можно сделать "горячим",
то есть осуществлять переход при нажатии на
изображение:</P>
<P><A HREF="index.html"><IMG SRC="picture.gif">
</A></P>
</BODY>
</HTML>
```

Цветовая гамма HTML-документа

Цветовая гамма HTML-документа определяется атрибутами, размещенными внутри метки <BODY>. Вот список этих атрибутов:

bgcolor

Определяет цвет фона документа.

text

Определяет цвет текста документа.

link

Определяет цвет выделенного элемента текста, при нажатии на который происходит переход по гипертекстовой ссылке.

vlink

Определяет цвет ссылки на документ, который уже был просмотрен ранее.

alink

Определяет цвет ссылки в момент, когда на нее указывает курсор мыши и нажата ее правая кнопка, то есть непосредственно перед переходом по ссылке.

Цвет кодируется последовательностью из трех пар символов. Каждая пара представляет собой шестнадцатиричное значение насыщенности заданного цвета одним из трех основных цветов (красным, зеленым и синим) в диапазоне от нуля (00) до 255 (FF). Разберем несколько примеров.

bgcolor=#FFFFFF

Цвет фона. Насыщенность красным, зеленым и синим одинакова — FF (это шестнадцатиричное представление числа 255). Результат — белый цвет.

text=#000000

Цвет текста. Насыщенность красным, зеленым и синим одинакова — 00 (ноль). Результат — черный цвет.

link=#FF0000

Цвет гипертекстовой ссылки. Насыщенность красным — FF (255), зеленым и синим — 00 (ноль). Результат — красный цвет.

Кроме того, метка <BODY> может включать атрибут background="[имя файла]", который задает изображение, служащее фоном для текста и других изображений. Как и любое другое изображение, фон должен быть представлен в формате GIF (файл с расширением *.gif) или JPEG (файл с расширением *.jpg или *.jpeg).

Браузеры заполняют множественными копиями изображения-фона все пространство окна, в котором открыт документ, подобно тому, как при строительстве большие пространства стен покрывают маленькими (и одинаковыми) плитками.

Важно отметить, что цвет фона и изображение-фон никак не отображаются на бумаге при выводе HTML-документа на печать. Из этого есть одно важное практическое следствие: старайтесь не использовать текст белого цвета.

```
<HTML>
<HEAD>
<TITLE>Пример 9</TITLE>
```

```
</HEAD>
<BODY bgcolor=#FFFFFF text=#000000 link=#FF0000>
<H1>Слайд-демонстрация цветовых гамм </H1>
<P>Черный текст на белом фоне </P>
</BODY>
</HTML>
```

Таблицы

До настоящего времени мы имели дело с документами, в которых существовал только один "поток" текста. На практике иногда очень хочется расположить текст в несколько колонок. Таблица может в этом помочь.

```
<HTML>
<HEAD>
<TITLE>Пример 10</TITLE>
</HEAD>
<H1>Простейшая таблица </H1>
<TABLE BORDER=1> <!--Это начало таблицы-->
<CAPTION> <!--Это заголовок таблицы-->
У таблицы может быть заголовок
</CAPTION>
<TR> <!--Это начало первой строки-->
<TD> <!--Это начало первой ячейки-->
Первая строка, первая колонка
</TD> <!--Это конец первой ячейки-->
<TD> <!--Это начало второй ячейки-->
Первая строка, вторая колонка
</TD> <!--Это конец второй ячейки-->
</TR> <!--Это конец первой строки-->
<TR> <!--Это начало второй строки-->
<TD> <!--Это начало первой ячейки-->
Вторая строка, первая колонка
</TD> <!--Это конец первой ячейки-->
<TD> <!--Это начало второй ячейки-->
Вторая строка, вторая колонка
</TD> <!--Это конец второй ячейки-->
</TR> <!--Это конец второй строки-->
</TABLE> <!--Это конец таблицы-->
</BODY>
</HTML>
```

Таблица начинается с метки <TABLE> и заканчивается меткой </TABLE>. Метка <TABLE> может включать несколько атрибутов:

ALIGN

Устанавливает расположение таблицы по отношению к полям документа. Допустимые значения: ALIGN=LEFT (выравнивание влево), ALIGN=CENTER (выравнивание по центру), ALIGN=RIGHT (выравнивание вправо).

WIDTH

Ширина таблицы. Ее можно задать в пикселах (например, WIDTH=400) или в процентах от ширины страницы (например, WIDTH=80%).

BORDER

Устанавливает ширину внешней рамки таблицы и ячеек в пикселах (например, BORDER=4). Если атрибут не установлен, таблица показывается без рамки.

CELLSPACING

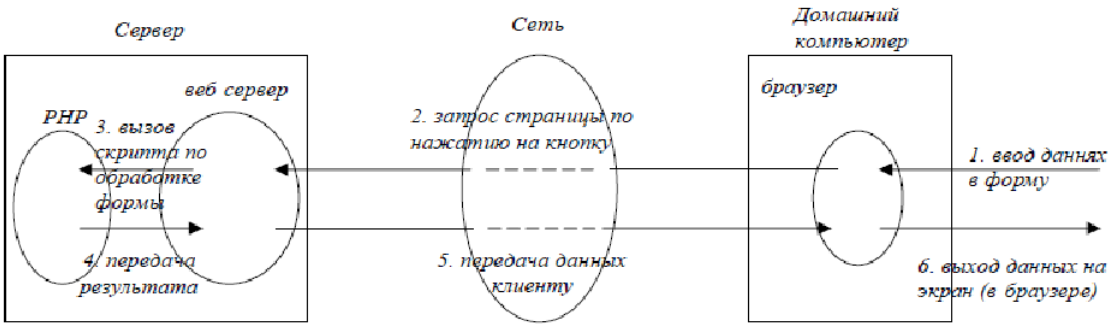
Устанавливает расстояние между рамками ячеек таблицы в пикселах (например,

CELLSPACING=2).
CELLPADDING
Устанавливает расстояние между рамкой ячейки и текстом в пикселах (например, CELLPADDING=10).
Таблица может иметь заголовок (<CAPTION> ... </CAPTION>), хотя заголовок не является обязательным. Метка <CAPTION> может включать атрибут ALIGN. Допустимые значения: <CAPTION ALIGN=TOP> (заголовок помещается над таблицей) и <CAPTION ALIGN=BOTTOM> (заголовок помещается под таблицей).
Каждая строка таблицы начинается с метки <TR> и заканчивается меткой </TR>. Метка <TR> может включать следующие атрибуты:
ALIGN
Устанавливает выравнивание текста в ячейках строки. Допустимые значения: ALIGN=LEFT (выравнивание влево), ALIGN=CENTER (выравнивание по центру), ALIGN=RIGHT (выравнивание вправо).
VALIGN
Устанавливает вертикальное выравнивание текста в ячейках строки. Допустимые значения: VALIGN=TOP (выравнивание по верхнему краю), VALIGN=MIDDLE (выравнивание по центру), VALIGN=BOTTOM (выравнивание по нижнему краю).
Каждая ячейка таблицы начинается с метки <TD> и заканчивается меткой </TD>. Метка <TD> может включать следующие атрибуты:
NOWRAP
Присутствие этого атрибута означает, что содержимое ячейки должно быть показано в одну строку.
COLSPAN
Устанавливает "размах" ячейки по горизонтали. Например, COLSPAN=3 означает, что ячейка простирается на три колонки.
ROWSPAN
Устанавливает "размах" ячейки по вертикали. Например, ROWSPAN=2 означает, что ячейка занимает две строки.
ALIGN
Устанавливает выравнивание текста в ячейке. Допустимые значения: ALIGN=LEFT (выравнивание влево), ALIGN=CENTER (выравнивание по центру), ALIGN=RIGHT (выравнивание вправо).
VALIGN
Устанавливает вертикальное выравнивание текста в ячейке. Допустимые значения: VALIGN=TOP (выравнивание по верхнему краю), VALIGN=MIDDLE (выравнивание по центру), VALIGN=BOTTOM (выравнивание по нижнему краю).
WIDTH
Устанавливает ширину ячейки в пикселах (например, WIDTH=200).
HEIGHT
Устанавливает высоту ячейки в пикселах (например, HEIGHT=40).
Если ячейка таблицы пуста, вокруг нее не рисуется рамка. Если ячейка пуста, а рамка нужна, в ячейку можно ввести символьный объект (non-breaking space — неразрывающий пробел). Ячейка по-прежнему будет пустой, а рамка вокруг нее будет.
Любопытно отметить, что любая ячейка таблицы может содержать в себе другую таблицу.
2.2 Перечень используемого оборудования
2.2.1 Персональный компьютер
2.2.2 Описание практической работы
3 Задание
3.1 Создайте файл my.html со своими данными: имя, фамилия отчество, группа, специальность. Используйте разные шрифты и выравнивания
3.2 В файл my.html добавьте три горизонтальных линии после фамилии, имени, отчества.

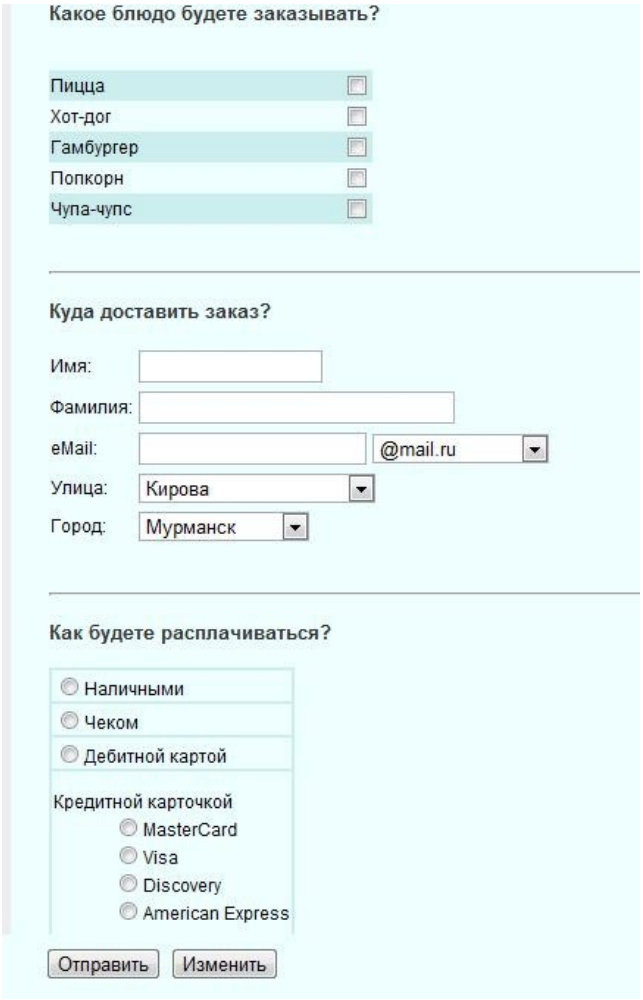
Линии должны иметь разный тип. После этого введите столбиком, изучаемые предметы этой сессии.
3.3 Измените в своем файле начертание (стиль шрифта) группы, факультета, название предметов
3.4 Откройте свой файл и преобразуйте список предметов в нумерованный список. Создайте маркированный список своих интересов (хобби), сопроводив его заголовком.
3.5 Создайте другой файл, назовите его zadan.html. Откройте свой файл и создайте в нем ссылку с именем «задания по лабораторной работе». В файл zadan.html укажите все задания, которые перечислены в данной практической работе.
3.6 Найдите на компьютере графические файлы и включите их в состав своего файла my.html.
3.7 Используя разные цветовые гаммы определите цвет фона и цвет каждого шрифта, используемого в документе.
3.8 Создайте в своем документе расписание занятий на текущую неделю.
4 Контрольные вопросы
4.1 Как форматировать текст по ширине?
4.2 Чтобы убрать синюю рамку вокруг картинки-ссылки?
4.3 Обязательно ли использовать кавычки в значениях атрибутов?
4.4 Как вставлять комментарии в HTML?
Требования к оформлению отчетного материала:
5 Содержание отчета
5.1 Наименование работы
5.2 Цель работы
5.3 Задание
5.4 Выводы по работе
5.5 Ответы на контрольные вопросы
Требования к оформлению отчетного материала: отчет о выполненной работе предоставлен с кодами и скриншотами веб-страниц; отчет содержит штамп
Форма контроля: отчет
Ссылки на источники: например: [2,3]

Практическая работа № 3
«Создание формы на html-странице»

Количество часов на выполнение:6, из них на практическую подготовку 6
Оборудование: Windows, Notepade++, Chrome, Mozilla FireFox
Задание:
Формы предназначены для организации взаимодействия с пользователем. Они позволяют вводить текст, осуществлять выбор из предложенных значений при помощи списков или кнопок, организовывать интерактивный обмен информацией между Web-страницей и сервером. Как правило, форма работает совместно с установленным на сервере сценарным приложением, обрабатывающим введенную информацию. Механизм обработки форм представлен на рисунке



Все формы начинаются тэгом <form> и завершаются</form>. Формы нельзя вкладывать одну в другую!!!
Структура формы такова:
<form method = "get | post" action = "URL" enctype= "MIME"> Элементы формы и все



остальное.... </form> method
Метод посылки сообщений из данной формы. Посылать данные можно двумя способами:
- GET: Информация из формы добавляется в конец URL. который был указан в описании за головка формы.

Пример: http://www.webboard.ru/wb.php?board=I0767
Этот метод рекомендуется для обмена небольшими (до 256 байт) порциями данных, а также для передачи данных в другой HTML-документ, который может с помощью JavaScript их обработать на стороне клиента.
- POST: Данный метод передает всю информацию о форме немедленно после обращения к указанному URL. CGI-программа на сервере получает данные формы из стандартного потока ввода. Данный метод рекомендуется к использованию на стороне сервера.

enctype
Указание типа передаваемой информации. В зависимости от MIME-типа информация будет преоб¬разована соответствующим образом.
action
Описывает URL. который будет вызываться для обработки формы. Данный URL почти всегда ука¬зывает на CGI-программу, обрабатывающую данную форму. Если данные надо принять по электронной почте, то необходимо ввести точный адрес почтового ящика.
Пример: action = "mailto:xxx@xxx.xx enctype=""text/plain"
Элементы формы
<INPUT > - ввод элемента формы
Атрибуты тэга
type = text | radio | submit -тип поля name = myName - имя поля (обязательный атрибут!!!)
Типы полей ввода:
<input type="TEXT" >: однострочное поле ввода.
Атрибуты тэга
size=20 - размер отображаемого поля ввода на экране (20- по умолчанию)
maxlength=4 - максимальная длина вводимого значения в символах
Пример: введите имя: <input name="имя" type="text" size="40">
 введите серийный код: <input name="код" type="text" size="20" maxlength="10">
<input type= "RESET">: кнопка сброса.
Данный тип обозначает кнопку, при нажатии которой все поля формы примут значения, описанные для них по умолчанию (или очистка формы).
<input type= "SUBMIT">: кнопка для отправки формы.
По щелчку, будет вызываться почтовая программа OutLook и форма отправляется по адресу (URL), указанному в параметре ACTION. Атрибут VALUE может содержать надпись на кнопке.
<input type= "RADIO"> : одиночный выбор значения из нескольких (радиокнопка). Для создания набора альтернатив вам необходимо создать несколько полей ввода с атрибутом TYPE = "RADIO" с разными значениями атрибута VALUE, но с одинаковыми значениями атрибута NAME. В CGI-программу будет передано значение типа NAME = VALUE, причем VALUE примет значение атрибута VALUE того поля ввода, которое будет выбрано. Выбор одного из полей автоматически отменяет выбор всех остальных полей того же имени (атрибут NAME).
Атрибуты тэга
name="myName"- имя возвращаемой переменной value="yes"- значение возвращенной переменной (т. е. "yes") checked будет выбрано по умолчанию
Пример:
<input type="radio" name="Мои друзья" value="Ed"checked>Эдуард <input type="radio" name="Мои друзья"value="Anton">Антон <input type="radio" name="Мои друзья"value="Oleg">Олег <input type= "CHECKBOX"> : множественный выбор.
Пример:
<input type="cheekbox" name="Мои друзья" value="Ed"checked>Эдуард <input type=" checkbox" name="Мои друзья"value="Anton">Антон <input type=" checkbox" name="Мои друзья"value="Oleg">Олег
<input type= "PASSWORD">: одиночный для ввода пароля. То же самое, что и атрибут TEXT, но вводимое пользователем значение не отображается браузером на экране
Атрибуты тэга
size =20 - размер отображаемого поля ввода на экране (20- по умолчанию) maxlength=4 - максимальная длина вводимого значения в символах
<input type= "BUTTON">: кнопка. Простая кнопка. Параметр "value = " задает надпись на кнопке. Используется для вызова программы на JavaScript, например, для проверки

данных формы.
<input type= "FILE">: передача файлов при помощи форм. В параметрах <FORM> необходимо указать ENCTYPE = "multipart/form-data". Внимание! Данная возможность требует поддержки получения файлов Web-сервером.
<SELECT > - раскрывающийся список Закрывающийся тэг - обязателен. Выбор значений из раскрывающегося списка значений, заданных при помощи <OPTION> Возможен одиночный или множественный выбор.
Атрибуты тэга size = "5" - начальная высота списка multiple - разрешить множественный выбор нескольких значений
Пример: <select name="fruir" size="1"> <option value="яблоко">яблоко</option> <option value="банан"> банан </option> <option value="груша"> груша </option> <option value="лимон"> лимон </option> <option value="киви"> киви </option> <option value="папайя"> папайя </option> </select>
Для множественного выбора используйте select multiple.
<TEXTAREA > - многострочное текстовое поле.
Закрывающийся тэг - обязателен. Пример: <textarea name="address" rows="5" cols=25"> УФА ул. Р Зорге д. 12/2 офис 438. Тел.24-33-74 </textarea>
Атрибуты тега name = "address"- имя поля ввода rows = 5- высота поля в строках cols= 25 - ширина поля в символах wrap = off | virtual | physical - неразбивать строки -разбивать при вводе, но передавать как
- одну строку - разбивал, на строки и передавать также. Пример формы

```
<html>
 <head> <title>Формы 1</title>
<style>
 #a{ border : 2 dotted #6699cc;
 background : #ffffff;
color : Purple;
 }
</style>
</head>
<body>
<form action="mailto:psih_odinochka@fromru.coin?subject= Анкета с сайта"
name="anketa"enctype="text/plain" method="post">
<table border="4" cellspacing="0" cellpadding="2" bordercolorIight="#FF8080"
bordercolordark="#FFOOOO">
<colgroup bgcolor="#990000" style="color:white">
<tr>
<td>Имя::</td>
<td><input type="Text" size="40" name="l" maxlength="12" id="a"></td>
</tr><tr>
<td>Фамилия</td>
<td><input type="Text" size="40" name="2" id="a"></td>
</tr><tr>
 <td>Сайт< /td>
<td><input type="Text" name="3" value="http://www." id="a"></td>
</tr><tr>
<td>Пол</td>
<td><input type="Radio" name="4" va1ue="мужской">М
<input type="Radio" name="4" va1ue="женский">Ж </td>
</tr><tr>
<td  Образование</td>
<td><select name="5" styIe="background-color:#cc3333;" style="color:white">
```

```
<option va1ue="среднее">среднее
<option va1ue="высшее">высшее
 <option va1ue="незаконченное высшее">незаконченное высшее
</select></td>
</tr><tr>
<td> О себе<br> кратко</td>
<td><textarea rows="4" cols="40" name="6" id="a">тут  ввести текст</textarea></td>
</tr><tr >
<td>Я увлекаюсь</td>
<td><input type="Checkbox" name="7">Музыка
    <br><input type="Checkbox" name="8" checked>Литература
        <br><input type="Checkbox" name="9">Комп
      <br><input type="Checkbox" name="10">Internet </td>
</tr><tr>
<td><input type="Reset" value="очистить"></td>
<td><input type="Submit" value="отправить форму"> </td>
</tr>
</table>
</form>
</body>
</html>
```

2.2 Перечень используемого оборудования

2.2.1 Персональный компьютер

2.2.2 Описание практической работы

2.2.3 Электронный носитель информации

3 Задание

3.1 Создать web-сайт (Задание выбирается в соответствии с номером списка группового журнала), с применением обязательных элементов

№	Название сайта (тематика)	Кол-во страниц	Обязательные элементы
1	Личный сайт (страница успеваемости, хобби)	3	Таблица, маркированный список, цвет фона и шрифта, ссылки, форма для отзыва
2	Сайт туристической фирмы	3	Маркированные и нумерованные списки, рисунки, ссылки, форма заказа путевок
3	Сайт Интернет магазина	4	Форма счета, форма анкеты, ссылки, нумерованные списки
4	Сайт библиотеки	3	Таблица, форма заполнения формуляра, ссылки, шрифты, нумерованные списки
5	Сайт новостей	4	Маркированные списки, нумерованные списки, ссылки, форма заполнения рейтинга, цвет текста
6	Сайт погоды	3	Таблицы, маркированные списки, ссылки, рисунки, цвет гиперссылок, форма ввода данных о погоде
7	Сайт спортивного клуба	4	Форма вступления, таблицы, ссылки, нумерованные списки, цвет текста
8	Сайт выставочного комплекса	3	Форма отзывов, таблицы, маркированные списки, нумерованные списки, ссылки,
9	Сайт концертного зала	3	Маркированные списки, ссылки, цвет фона, цвет и формат шрифта, форма покупки билетов

10	Сайт инструментального ансамбля	3	Рисунки, шрифты, таблицы, ссылки, форма отзывов (гостевая книга)
11	Сайт железнодорожного вокзала	4	Таблицы, нумерованные списки, ссылки, форма покупки билета, рисунки
12	Сайт продажи сотовых телефонов	4	Нумерованные и маркированные списки, ссылки, цвета и шрифты, таблицы, форма заказа телефона
13	Сайт фотоателье	3	Рисунки, цвета, шрифты, ссылки, таблицы, форма заказа
14	Сайт транспортной компании	4	Рисунки, таблицы, нумерованные списки, ссылки, форма путевого листа
15	Сайт поликлиники	3	Регистрационная форма, ссылки, маркированные списки, таблицы
16	Сайт института	4	Ссылки, нумерованные списки, форма для абитуриента, цвета, шрифты
17	Сайт центрального рынка	3	Таблицы, маркированные и нумерованные списки, ссылки, шрифты, рисунки, форма отзыва
18	Сайт ресторана	4	Форма заказа, таблицы, нумерованные списки, цвета, ссылки
19	Сайт парка культуры и отдыха	3	Нумерованные и маркированные списки, ссылки, шрифты, рисунки, форма отзывов и предложений
20	Сайт журнала	4	Подписная форма, таблицы, маркированные списки, цвет ссылок, ссылки
21	Сайт коммерческой фирмы	3	Маркированные и нумерованные списки, рисунки, ссылки, форма заказа товара
22	Сайт Интернет аукциона	4	Форма заявки, форма анкеты, ссылки, нумерованные списки
23	Сайт архива	3	Таблица, форма заполнения формуляра, ссылки, шрифты, нумерованные списки
24	Сайт трамвайного маршрута	3	Ссылки, таблицы, форма путевого листа, шрифты, рисунки
25	Сайт web-студии	4	Подписная форма, таблицы, маркированные списки, цвет ссылок, ссылки
26	Сайт кадрового агентства	4	Таблица, форма заполнения формуляра, ссылки, шрифты, нумерованные списки
27	Сайт кафе	4	Форма заказа, таблицы, нумерованные списки, цвета, ссылки
28	Сайт газеты	4	Подписная форма, таблицы, маркированные списки, цвет ссылок, ссылки
29	Сайт молодежной одежды	4	Таблицы, маркированные и нумерованные списки, ссылки, шрифты, рисунки, форма заказа
30	Сайт театра	4	Маркированные списки, ссылки, цвет фона, цвет и формат шрифта, форма покупки билетов

3.2 Сохранить сайт на электронный носитель

4 Контрольные вопросы

4.1 Какие цвета вы использовал и при создании web-сайта?

4.2 Какой вид сайта вы создали?
4.3 Какие типы графических файлов вы использовали?
4.4 Как задаются списки переключателей?
4.5 С помощью каких тегов создаётся форма с использованием упорядоченных списков для нумерации полей ввода?
4.6 Для чего необходим тег <FORM>?
Требования к оформлению отчетного материала:
5 Содержание отчета
5.1 Наименование работы
5.2 Цель работы
5.3 Задание
5.4 Выводы по работе
5.5 Ответы на контрольные вопросы
Требования к оформлению отчетного материала: отчет о выполненной работе предоставлен с кодами и скриншотами веб-страниц; отчет содержит штамп
Форма контроля: отчет
Ссылки на источники: например: [2,3]

Практическая работа № 4

«Форматирование web-страниц с использованием каскадных таблиц стилей»

Количество часов на выполнение:10, из них на практическую подготовку 10
Оборудование: Windows, Notepade++, Chrome, Mozilla FireFox
Задание:
1 Цель: Изучить основы технологии CSS на примере разработки собственного сайта.
2 Пояснения к работе
2.1 Краткие теоретические сведения
1. Каскадные таблицы стилей
Стиль — это набор правил оформления элемента web-стра¬ницы.
Селектор — элемент стиля, в селекторе указаны параметры форматирования (цвет элемента, фоновый цвет, гарнитура шриф¬та, размер символов, отступы, стили границ и др).
Типы селекторов: селекторы тегов, классы и идентификаторы.
Синтаксис стиля:
селектор {параметр].: значение;
параметр2: значение]., значение2;
}
При написании стиля можно использовать любой регистр клавиатуры, пробелы необязательны, значение можно помещать в двойные кавычки, значок // обозначает начало комментария.
Расположение стиля.
1. В отдельном файле, если одинаковым стилем оформлены разные страницы. Стили записываются в текстовый файл с рас¬ширением CSS (например, mystyle.css):
Пример:
p { color: gold;
background-color: #990000;
}
В коде страницы, на которой используется стиль из этого файла, делают ссылку: Пример:
<head>
<link rel="stylesheet" type="text/css"
href="mystyle.css">
</head>
Стиль применяется, когда используют тег, оформленный данным стилем:

<body>
текст
</body>
2. В разделе head в теге style, если стиль на странице исполь¬ зуется несколько раз или в разных тегах.
Пример:
<head>
<style type="text/css"> p { color: gold; background-color: #990000;
}
</style> </head>
3. В любом теге в атрибуте style, если стиль используется редко. Пример:

текст
Селекторы тегов. Любой тег HTML может быть селектором. Синтаксис:
Тег { Параметр: Значение; }
Пример:
а) <style>
body { text-align: justify; color: black; font-family:
Arial; }
</style>
б) <headxstyle>
HI { font-family: Arial, Helvetica, Verdana,
sans-serif; font-size: 150%; font-weight: light }
</style></head>
<body>
<H1>Заголовок</H1> Обычный текст
</body>
Классы. Классы используют, когда нужно применить стиль к разным тегам web-страницы: ячейкам таблицы, ссылкам, па¬раграфам и др.
Синтаксис класса:
Тег.Имя класса { Параметр: Значение; }
В тег добавляется атрибут class= "Имя класса". Пример:
<headxstyle>
P.cite (color: navy; font-size: 80% margin: 20px }
// параграф с классом cite
</stylex/head>
<body>
 текст
TeKCT
</body>
Если для класса не указывать конкретный тег, тогда класс можно применять к любому тегу. Синтаксис класса:
.Имя класса { Параметр: Значение; }
Пример:
<headxstyle>
.mycolor { color: navy; color-background: yellow; }
</style></head>
<h2 class=mycolor>TeKCT</h2>
<tablextr>
<td class=mycolor> текст </td>
<td>TeKCT</td>
</tr></table>

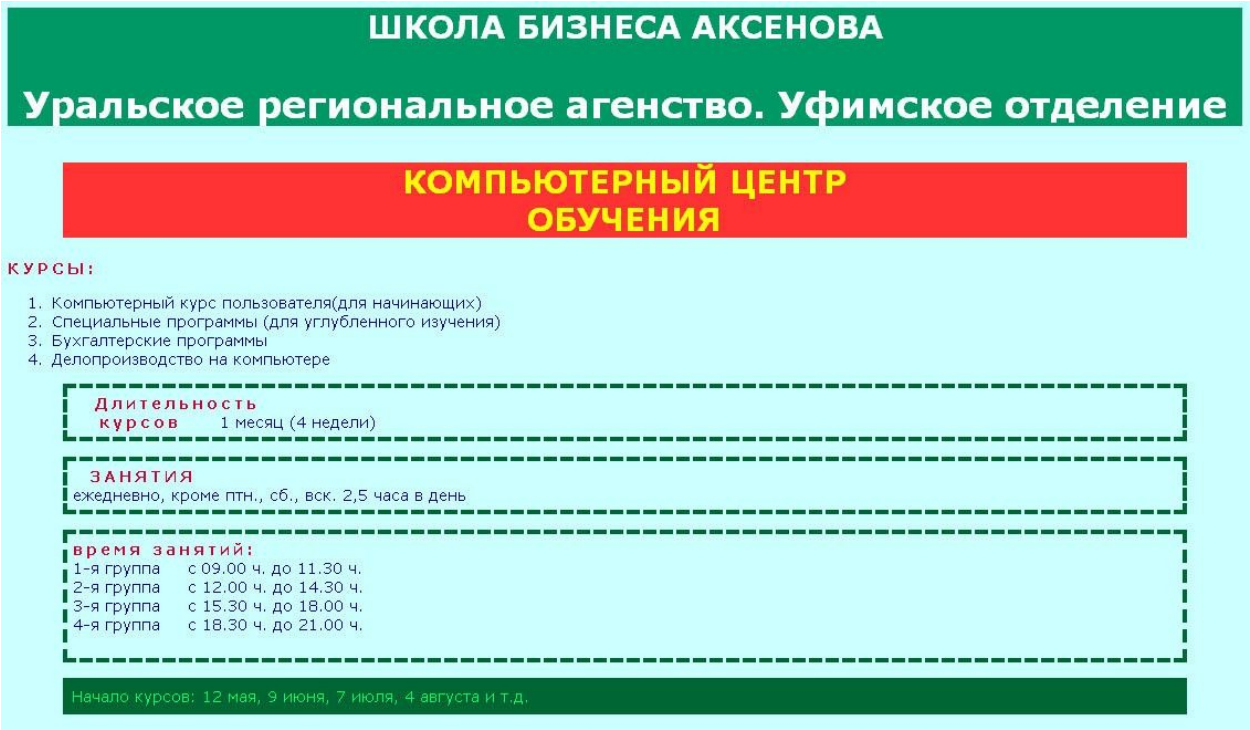
Тег span используют для создания выделенного текста, бук¬виц, цитат и др. Пример:
<headxstyle>
.capital { font-size: 150%; color: red; }
</styleX/head>
<body>
ByKBnua
</body>
Псевдоклассы. Псевдокласс используется только для оформ¬ления ссылок. Виды псевдоклассов: непосещенная ссылка (link), состояние ссылки под курсором мыши (hover), состояние ссылки в момент щелчка (active), посещенная ссылка (visited).
Синтак¬сис псевдокласса: A:псевдокласс { Параметр: Значение; }
Пример:
<headxstyle type="text/css">
a:link { color: #003366; }
a:visited { color: #660066; }
a:hover { color: #800000; }
a:active (color: #FF0O0O; }
</stylex/head>
<body>
| CcbinKa K/a> I Ссылка 2 I
CcbtnKa 3 I
</body>
Пример:
<head> <style>
a.linkl { font-size: 12px; color: green }
a.linkl:hover { color: red }
a.link2 {font-size: 14px; color: blue }
a.Iink2:hover { color: red }
</style> </head>
<body link=#0000ff>
I CcbinKa 1 | <a href=link2.html
class=linkl>CcbmKa 2 | <a href=link3.html
class=link2>CcbLnKa 3 |
</body>
Пример использования наиболее востребованного псевдо¬класса (hover):
<headxstyle type="text/css">
a {color: #003366;} a:hover {color: #800000;}
</style></head>
Пример использования различных классов для разных видов ссылок:
<head> <style>
a.linkl {font-size: 12px; color: green;}
a.linkl:hover { color: red;}
a.link2 {font-size: 14px; color: blue;}
a.Iink2:hover {color: red;}
</style> </head>
<body>
I CcbuiKa 1 | <a href=link2.html
class=linkl>CcbmKa 2 | <a href=link3.html
class=link2>CcbuiKa 3 |
+ </body>
Шрифт.
1. Семейства шрифтов font-family

Таблица 4.1. Примеры шрифтов, входящих в различные семейства шрифтов.Требования к оформлению отчетного материала:

Семейство шрифтов	Примеры шрифтов	Описание семейства
serif	Times New Roman, Book Antiqua	Шрифт с засечками
sans-serif	Helvetica, Arial, Verdana	Шрифт без засечек
cursive	Zapf-Chancery	Курсивный шрифт
fantasy	Western, Comic	Декоративный шрифт
monospace	Courier New	Моноширинный шрифт

Варианты шрифтов разделяются запятыми, название шриф¬та, состоящее из нескольких слов, заключается в кавычки:
p {font-family: 'Western, Times New Roman', serif;}
2. Толщина шрифта font-weight: lighter, bold, bolder. Пример:
p.bold {font-weight: bold;} (относительно предыдущего текста).
3. Размер шрифта font-size: относительный (проценты — %), или абсолютный (пикселы — px, пункты — pt, сантиметры — cm и миллиметры — mm). Пример:
hi {font-size: 200%;} (относительно основного текста)
h2 {font-size: 50px;}
h3 {font-size: 12pt; }
4. Оформление текста text-decoration. Значения: подчеркива¬ние (underline), надчеркивание (overline), зачеркивание (line-through), нет подчеркивания (попе).
Примеры:
h4 {text-decoration: underline;}
p {text-decoration: overline;}
.wrong {text-decoration: line-through;}
a {text-decoration: none;}
Цвет.
1. Цвет элемента color — цвет элемента в модели rgb или по стандартным названиям цветов.
Примеры:
hi {color: yellow;}
hi {color: yellow;}
.active {color: #FFFF00;}
2. Цвет фона background-color — цвет фона элемента (фоно¬вый цвет).
Пример:
hl.grayback {background-color: #CCCCCC;}
Отступы.
1. Выравнивание text-align — выравнивание абзаца относи¬тельно страницы или ячейки таблицы. Значения: по левому краю (left), по центру (center), по правому краю (right), по шири¬не (justify).
Пример:
p {text-align: justify;}
2. Отступ первой строки text-indent — отступ первой строки абзаца (красная строка). Значение задается в пикселах (px) или в пунктах (pt).
Пример:
p {text-indent: 50pt;}
3. Межстрочные интервалы Hne-heighn — междустрочный ин¬тервал абзаца. Значение задается в пикселах (px), пунктах (pt) или в процентах от размера шрифта (%).
Пример:
p {line-height: 50%;} межстрочные интервалы.
4. Отступы вокруг элемента margin — отступы вокруг элемен¬та. Значение задается в пикселах (px) или в пунтках (pt). Различ¬ные типы этого параметра: одинаковый отступ со

всех сторон (margin), отступ слева (margin-left), отсуп справа (margin-right), отступ сверху (margin-top), отступ снизу (margin-bottom).
Примеры:
p {margin: ЗОpx;}
img {margin-left: 20px;}
Рамка.
1. Толщина рамки border-width — толщина границы. Различ¬ные типы этого параметра: толщина левой границы {border-left-width), толщина верхней границы {border-top-width), толщина правой границы {border-right-width), толщина нижней границы {border-bottom-width). Значение задается в пикселах (px).
Пример:
.lineTop {border-top-width: 3px;}
2. Цвет рамки border-color — цвет рамки (границы). Пример:
.lineRed {border-color: #FF0000;}
3. Стиль рамки border-style — стиль границы. Значения: гра¬ницы нет {попе), сплошная линия {solid), пунктирная линия{dotted), штриховая линия {dashed), двойная линия {double), объ¬емная канавка {groove), объемный гребень {ridge), объемнаякнопка внутрь {inset), объемная кнопка наружу {outset).
Пример:
.lineSimple {border-style: outset;}
4. Обобщающий стиль рамки border — обобщающий стиль границы, объединяющий три параметра: толщину, стиль и цвет границы. Порядок записи параметров неважен:
border: ширина стиль цвет; border-tor: ширина стиль цвет; border-right: ширина стиль цвет; border-bottom: ширина стиль цвет; border-left: ширина стиль цвет;
Пример:
td.blueBorder {border: 4px dotted #003399}
Обобщающий стиль можно задавать для определенной стороны границы: слева (border-left), сверху (border-top), справа (border-right), снизу (border-bottom).
Пример:
td.red21ine {border-color: #FF0000;
border-left: 2px double;}
2.2 Перечень используемого оборудования
2.2.1 Персональный компьютер
2.2.2 Описание практической работы
3 Задание
3.1 Отформатируйте страницу с помощью листа стилей, размещённого в теге <style>...</style> (см. пример) Сохраните страницу в личной папке в файл style1.html



2. Запишите исходный код таблицы стиля в отчет

4 Контрольные вопросы

4.1 Что является компонентами страницы HTML?

4.2 Какие существуют способы управления настройками браузера:

4.3 В чем главная особенность каскадных таблиц стилей?

4.4 Как сделать неподчеркнутые ссылки везде?

4.5 Как задать отступы содержимого странички от краев окна браузера?

4.6 Как задать фоновую картинку для ячейки таблицы?

4.7 Как сделать информацию о пользователе ICQ, чтобы около номера был показатель присутствия в сети?

Требования к оформлению отчетного материала:

5 Содержание отчета

5.1 Наименование работы

5.2 Цель работы

5.3 Задание

5.4 Выводы по работе

5.5 Ответы на контрольные вопросы

Требования к оформлению отчетного материала: отчет о выполненной работе предоставлен с кодами и скриншотами веб-страниц; отчет содержит штамп

Форма контроля: отчет

Ссылки на источники: например: [2,3]

Практическая работа № 5
«Вёрстка»

Количество часов на выполнение: 8, из них на практическую подготовку 8

Оборудование: Windows, Notepade++, Chrome, Mozilla FireFox

Задание:

Цель работы: Изучить возможность верстки веб-страницы с помощью тегов HTML5и CSS.

Блочная верстка – это верстка страницы с использованием слоев или блоков, в основе используются теги

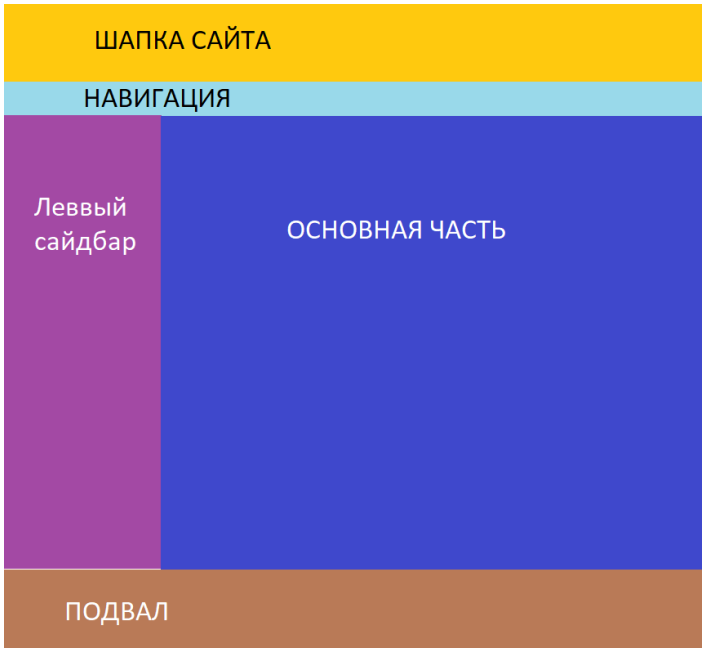
div.

Основной принцип такой верстки в том, что вся страница разделяется на блоки, они могут накладываться друг на друга, им можно задавать отступы с краев, в данные блоки помещаются остальные элементы страницы. Ко всем блокам применяются стили CSS, которые показывают браузерам размер, расположение, отступы снаружи и внутри блока и много другое.
Стили мы будем прописывать с использованием тега <style>в части <head>.

Ход работы

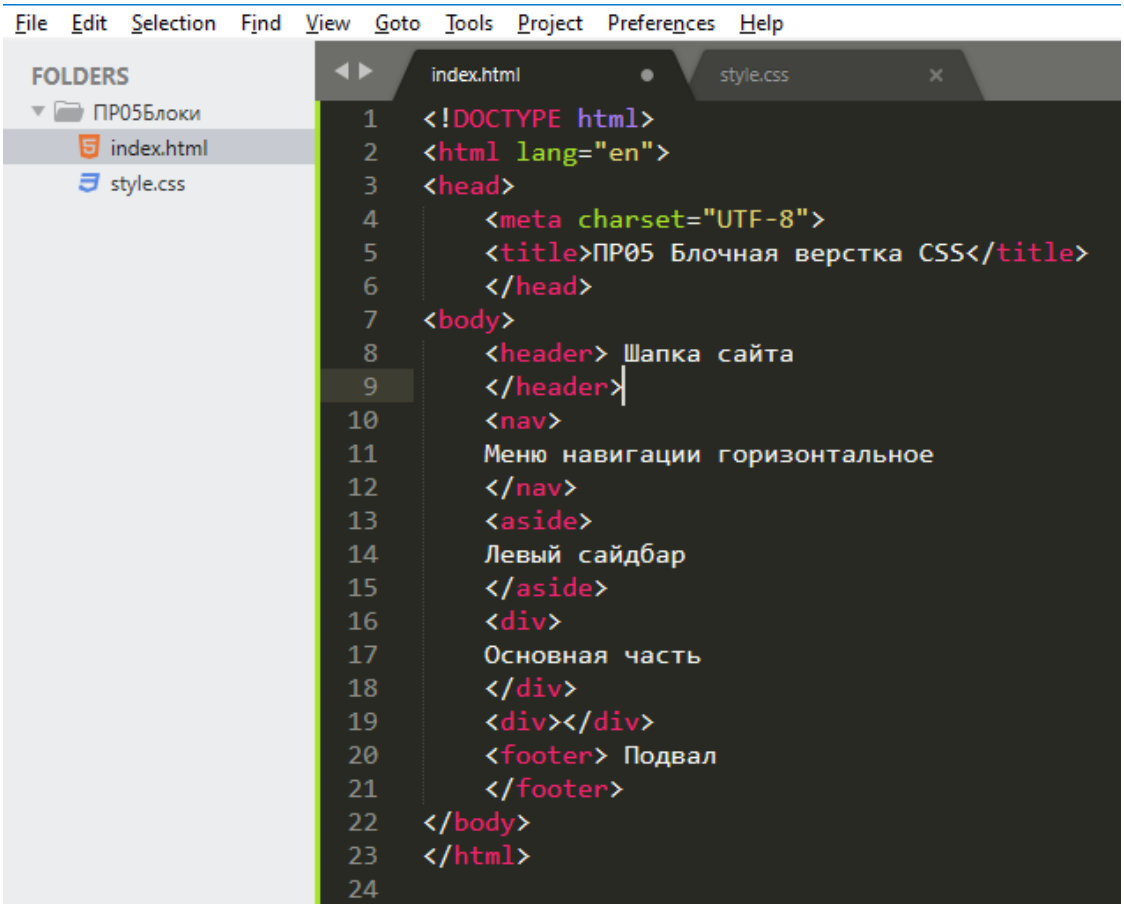
Для начала нам необходимо понять какая структура страницы нам нужна, то есть нарисовать ее структуру.

Мы будем использовать простую структуру, которая встречается на большинстве сайтов. Она будет состоять из: шапки, горизонтального меню, левого сайдбара, основной части и подвала.
Представим необходимую структуру графически:



После того как мы нарисовали структуру в графическом варианте мы можем определиться с тем какие теги мы будем использовать для создания структуры страницы с использованием HTML5.

Создайте структуру страницы:



Если мы посмотрим как выглядит наш код в браузере то увидим следующее

Шапка сайта
Меню навигации горизонтальное
Левый сайдбар
Основная часть
Подвал

На нужный результат совсем не похоже, кроме надписей. Для того чтобы наша страница приобретала нужный вид добавляем стили CSS в часть <head>.

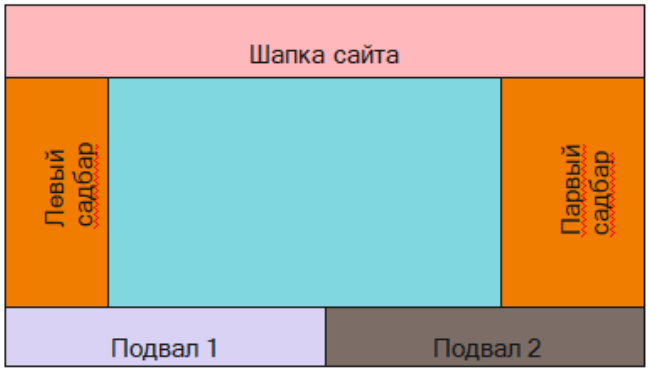
```
<style> body {
background: #f5f5f5;
color: #000000;
font-family: Arial, Times New Roman;
font-size: 16px;
}
header {
    background: yellow
    height: 100px;
    width: 100%;
}
nav {
    background: #330044; color: white;
    width: 100%; height: 50px;
}
aside {
    background: #1baf5d; float: left;
    width: 10%;

    height: 400px;
}
#content { background: gray; float: left;
width: 90%;
}
#clear { clear:both;

}
footer {
    background: #ff0404; color: white;
    height: 80px; width: 100%;
}
</style>
```

Проверим результат в браузере

После применения стилей видно что наш сайт уже стал приобретать вид того что нам нужно. Теперь разберем код стилей.
Итак, начнем по порядку с тега <header>нам необходимо задать ему ширину в 100% от окна, это делается с помощью свойства width:100%, и задаем высоту в 100 пикселей с помощью свойства height:100px. И соответственно используя свойство background: yellow. Шапка готова. Значения ширины и высоты могут принимать значения в пикселях и в процентах. Что использовать зависит от конкретных ситуаций. Переходим дальше. Следующим элементом у нас является меню навигации. При стилизации блока меню используются те же самые свойства что и при стилизации шапки. Плюс мы добавили свойство color: white, оно позволяет указать цвет текста. В качестве значений для свойств задающих цвета могут выступать названия цветов (например, red. Green, black), так и шестнадцатиричные значения цветов (например, #ff00ff, #000000 и т.д.).
Следующим блоком у нас является сайдбар созданный с помощью тега <aside>, к нему применяем свойства ширины, высоты, цвет заливки и добавляется свойство float:left, оно позволяет задать позиционирование блока по левому краю. При задании размеров блоков нужно правильно рассчитывать ширину блоков, которые находятся рядом. Вся ширина экрана это 100%, то есть для того чтобы нам разместить два блока рядом необходимо одному блоку задать ширину X второму задать ширину Y, но главное следует всегда просчитывать чтобы сумма значений X и Y всегда равнялась 100%. В противном случае может появиться пустое пространство, либо блоки не будут умещаться и будут не структурированы, то есть какой либо из блоков может переместиться ниже или выше другого, либо наложиться сверху.
Следующий блок это основная часть страницы, его мы создали с помощью тега и присвоили ему идентификатор #content . данному идентификатору мы создали стили заливки фона (background), ширину (width), высоту (height), и обтекаемость (float).
Последним блоком является подвал, который мы создали с помощью тега <footer>. Стили для данного блока практически идентичны стилям которые мы использовали при стилизации шапки страницы.
Это небольшой пример того как можно сверстать веб-страницу с помощью блоков и стилей CSS. Большие и качественные проекты безусловно требуют гораздо более сложного кода и структуры тегов. Но всегда нужно с чего-то начинать и данная работа поможет вам в этом.
Примечание: В случае отличий расположения блоков с образцом, выполните позиционирование. Используйте, например, для подвала свойство pozition:absolute; или pozition:fixed;

Практическое задание для выполнения

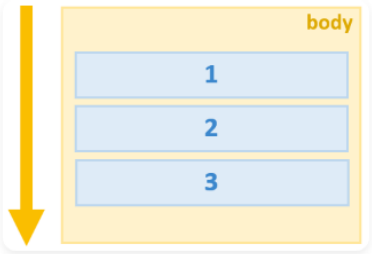
1) Вам необходимо сверстать предложенный макет страницы используя HTML5 и CSS. Стили необходимо присоединить с помощью внешнего файла style.css. Его необходимо разместить в той же папке что и ваш HTML файл.



2) Стилизуйте шрифты для текста на странице, добавьте границы к блокам и пропишите стили,
3) Создайте горизонтальное меню в шапке сайта или добавив блок навигация.
Меню: Главная Новости Контакты Обратная связь
4) Сделайте внутренние и внешние отступы в блоках.
5) Изучите Методы позиционирования элементов в CSS
Методы позиционирования элементов в CSS
Базовый поток документа
HTML-документ состоит из большого количества элементов, вложенных друг в друга. Чтобы из этих элементов и CSS построить изображение страницы, их необходимо как-то в ней расположить. По умолчанию размещение всех элементов на странице осуществляется в нормальном или базовом потоке.

Что это значит? Во-первых, вывод элементов на страницу браузер осуществляет в том порядке, в котором они следуют в HTML коде.

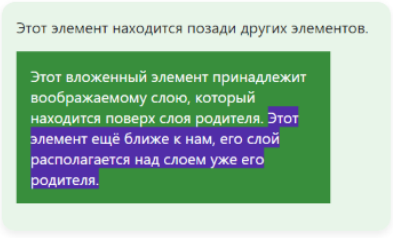
```
<body>
  <div>1</div>
  <div>2</div>
  <div>3</div>
</body>
```



Во-вторых, в коде элементы вложены друг в друга, и чтобы это учитывать при выводе используют так называемые воображаемые слои для отображения элементов. При этом слой элемента тем выше (ближе к нам), чем данный элемент является более вложенным в коде, т.е. глубже расположен в нём.
<body>
Этот элемент находится позади других элементов.

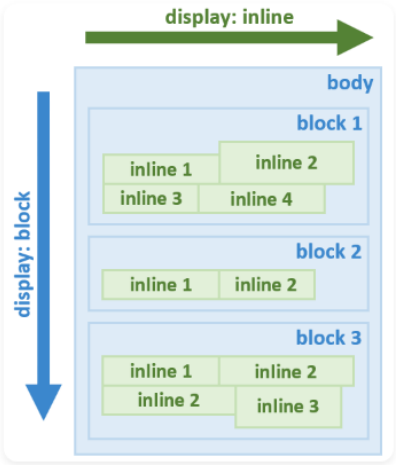
Этот вложенный элемент принадлежит воображаемому слою, который находится поверх слоя родителя.
Этот элемент ещё ближе к нам, его слой располагается над слоем уже его родителя.

</body>



В-третьих, положение элемента в потоке зависит от значения свойства display.

```
<body>
<div class="block-1"><span class="inline-1">inline 1</span><span class="inline-2">inline 2</span><span class="inline-3">inline 3</span><span class="inline-4">inline 4</span></div>
<div class="block-2"><span class="inline-1">inline 1</span><span class="inline-2">inline 2</span></div>
<div class="block-3"><span class="inline-1">inline 1</span><span class="inline-2">inline 2</span><span class="inline-3">inline 3</span></div>
</body>
```



Например, элементы, имеющее блочное отображение (display: block) отображаются в потоке как прямоугольные области, каждый из них на новой линии друг под другом сверху вниз.

Ширина элементов с блочным отображением по умолчанию равна доступной ширине родительского элемента, т.е. элемента, в который каждый из них непосредственно вложен. Высота их по умолчанию равна такой величине,
которой будет достаточно, чтобы отобразить весь контент, который находится в каждом из них.
Элементы со строчным отображением (display: inline) выводятся иначе. Они в отличии от блочных элементов не размещаются каждый на новой строке, а следуют друг за другом слево направо. Если пространство справа закончилось, то они переносятся на следующую строку, а не на новую линию как элементы с блочным отображением.

Кроме block, inline есть и другие варианты отображения элементов, но все они располагаются в базовом потоке документа.

В CSS есть свойства, с помощью которых элементы можно «вырвать» из основного потока документа и задать им другое положение вне базового потока элементов.

К этим свойствам относятся position и float.

CSS-свойство position
CSS свойство position — это одно из свойств с помощью которого можно изменить базовое поведение элементов в потоке. Другими словами, данное свойство позволяет «выдернуть» любой элемент из потока документа и разместить его в другом месте относительно окна браузера или других элементов на веб-странице.

Свойство position имеет 5 значений:

static (статичное позиционирование);
relative (относительное);
absolute (абсолютное);
fixed (фиксированное);
sticky (липкое).
static — это значение по умолчанию. Оно означает что элемент находится в базовом потоке.

Каждый элемент в потоке занимает определённую область. Но область элемента не всегда сохраняется за ним при его позиционировании.

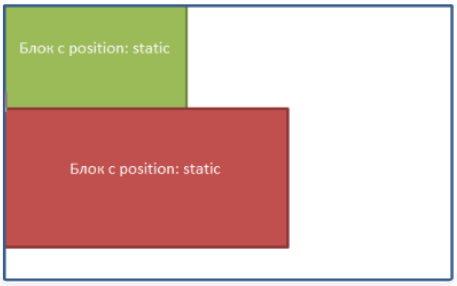
Это, например, происходит при задании элементу position: absolute или position: fixed. В этом случае место не сохраняется за элементом. Другие элементы его «не видят» и располагаются, игнорируя его присутствие в коде.

Статичное позиционирование (static)
Свойство position со значением static элементам назначается по умолчанию. Это значение означает что элемент является не позиционированным, т.е. отображается как обычно (в потоке).

Явная установка элементу CSS-свойства position: static может понадобиться только в том случае, когда нужно переопределить другое значение position установленное элементу.

Установка CSS свойств для задания положения элемента left, top, right и bottom никакого влияния на него не оказывают, т.к. его местонахождение определяется потоком документа. Пример выстраивания статично позиционированных элементов:

```
<body>
<div style="width: 200px; height: 100px; border: 1px solid black; background: green;"></div>
<div style="width: 300px; height: 150px; border: 1px solid black; background: red;"></div>
</body>
```



Относительное позиционирование (relative)
Установка относительного позиционирования элементу осуществляется посредством задания ему CSS свойства position: relative.

Относительно позиционированный элемент ведёт себя как элемент в потоке за исключением того, что его текущее положение можно при помощи определённых CSS свойств сместить. К этим CSS свойствам относятся left, top, right и bottom.

Например, для того чтобы элемент сдвинуть вверх или вниз относительного его исходного положения к нему нужно применить CSS свойство top или bottom:

```
position: relative;
/* для сдвига элемента вверх на 10px */
top: -10px; /* или bottom: 10px; */
```

/* для сдвига элемента вниз на 10px */
top: 10px; /* или bottom: -10px; */
Если одновременно установить top и bottom, то будет применено значение top, т.к. оно является более приоритетным, чем bottom:

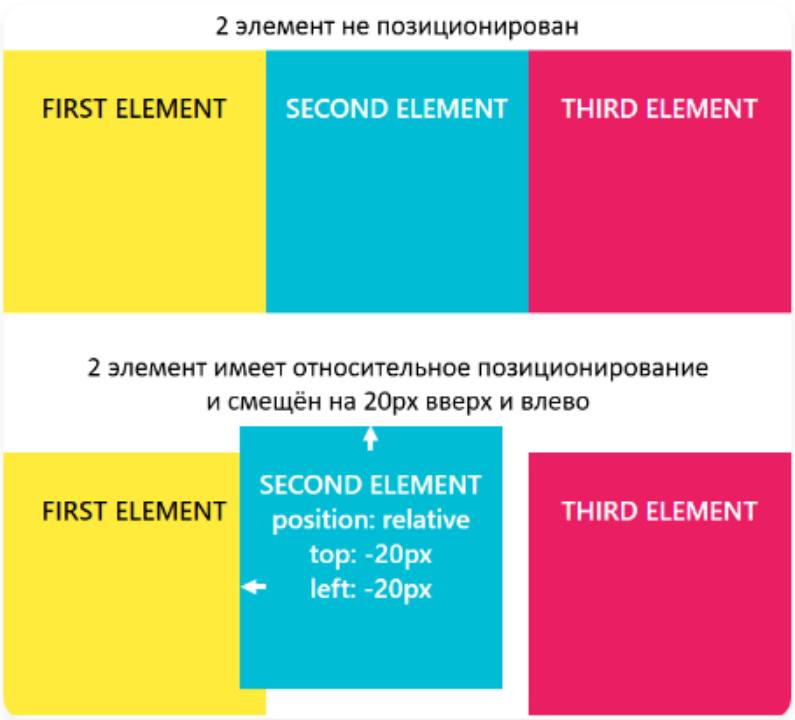
position: relative;
/* элемент или элементы, к которым применяется эти стили будут сдвинуты на 15px вверх, а не на 10px как указано в bottom */
top: -15px;
bottom: 10px;
Для сдвига элемента вправо или влево используется CSS свойство left или right:

position: relative;
/* для сдвига элемента влево на 20px */
left: -20px; /* или right: 20px; */
/* для сдвига элемента вправо на 20px */
left: 20px; /* или right: -20px; */
Если одновременно установить left и right, то приоритетным будет значение, находящееся в left:

position: relative;
/* элемент или элементы, к которым применяется эти стили будут сдвинуты на 25px вправо, т.к. значение left более приоритетно чем right */
left: 25px;
right: -20px;
Для сдвига по двум осям нужно использовать top или bottom, и left или right:

position: relative;
/* стили для сдвига элементов вверх и влево на 5px */
top: -5px;
left: -5px;
Пример, в котором 2 элементу установим относительное позиционирование и сместим его на 20px вверх и влево относительно его исходного положения:

  FIRST ELEMENT
  <!-- элемент имеет относительное позиционирование и смещён на 20px вверх и влево -->
     SECOND ELEMENT
  THIRD ELEMENT



Если в некоторой области страницы оказываются несколько позиционированных элементов, то они перекрывают друг на друга в определённом порядке. При этом по умолчанию выше оказывается тот элемент, который ниже описан в коде. Но порядок перекрытия элементов (их положение перпендикулярное экрану, т.е. вдоль оси Z) можно изменить. Осуществляется в CSS это с помощью свойства z-index. z-index может принимать отрицательные и положительные целые число, auto и 0. Но, хорошей практикой является использование в качестве z-index чисел из диапазона 0-9999.

При этом чем больше у элемента значение z-index, тем ближе он располагается к нам, и, следовательно, перекрывает все элементы в данной области, у которых значение z-index меньше.

Абсолютное позиционирование (absolute)
Установка абсолютного позиционирования элементу осуществляется посредством задания ему position: absolute.

Этот тип позиционирования позволяет разместить элемент именно там, где вы хотите.

Позиционирование выполняется относительно ближайшего позиционированного предка.

```
<div id="id-1" style="position: absolute">
  <div id="id-2" style="position: relative">
    <div id="id-3" style="position: absolute">
      ...
    </div>
  </div>
</div>
```

Под позиционированным элементом понимается элемент с position, равным relative, absolute, fixed или sticky.

В этом примере позиционирование элемента #id-3 будет выполнять относительно #id-2, т.к. он является позиционированным и является по отношению к нему более близким

предком.
Если данный элемент не был бы позиционированным, то позиционирование #id-3 выполнялось бы относительно #id-1:

```
<div id="id-1" style="position: absolute">
  <div id="id-2">
    <div id="id-3" style="position: absolute">
      ...
    </div>
  </div>
</div>
```

Если среди предков у элемента с position: absolute нет позиционированного элемента, то в этом случае он будет позиционироваться относительно HTML страницы, т.е. элемента body.

Когда элементу устанавливаем position: absolute без указания CSS-свойств (top, left, right и bottom), определяющих его положение, он будет находиться в том месте, в котором он был бы расположен, если бы находился в потоке (при этом при вычислении его положения учитываются только элементы, расположенные до него в коде и находящиеся в потоке).

При этом другие элементы его видеть не будут, и, следовательно, они будут расположены на странице, не обращая никакого внимание на него.

```
<div class="container">
  <div class="element-1">FIRST ELEMENT</div>
   <!-- элемент имеет абсолютное позиционирование и ему не установлены CSS-свойства top, bottom, left и right -->
  <div class="element-2" style="position: absolute;">SECOND ELEMENT</div>
  <div class="element-3">THIRD ELEMENT</div>
</div>
```

CSS-свойства для управления положением абсолютно позиционированного элемента работают по-другому чем с position: relative.

CSS-свойства top, bottom, left и right задают положение элемента относительно

ближайшего позиционированного предка или body, если такого предка нет.

Установить ширину (высоту) абсолютно позиционированному можно с помощью установки ему двух координат top и bottom (left и right).

Если элементу одновременно установить top, bottom и height, то предпочтение будет отдано top и height.

Абсолютное позиционирование применяется очень часто совместно с относительным позиционированием в дизайнерских целях, когда необходимо разместить различные элементы относительного друг друга, так же может применяться для создания выпадающих меню, разметки сайта и т.д.

```
<body>
<div style="width: 200px; height: 100px; border: 1px solid black; background: green;"></div>
<div style="width: 300px; height: 200px; position: absolute; top: 50px; left: 100px; border: 1px solid black; background:red;"></div>
</body>
```

Фиксированное позиционирование (fixed)
Задание элементу фиксированного позиционирования осуществляется посредством установки ему position: fixed.

Фиксированное позиционирование похоже на абсолютное, но в отличии от него оно всегда привязывается к краям окна браузера (viewport), и остаётся в таком положении даже при скроллинге страницы.

Фиксированное позиционирование применяется для закрепления на странице навигационных меню, кнопки «вверх», панелей с социальными кнопками и многого другого.

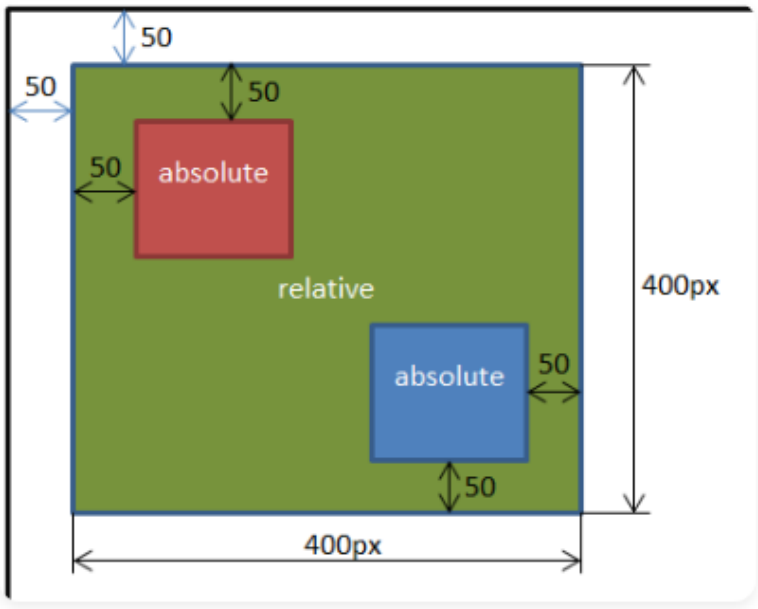
```
<body>
    <div style="width: 200px; height: 2000px; border: 1px solid black; background: green;"></div>
  <div style="width: 600px; height: 200px; position: fixed; top: 100px; left: 100px; border: 1px solid black; background:red;"></div>
</body>
```

Совместное использование относительного и абсолютного позиционирования
Относительное позиционирование очень часто используется вместе с абсолютным позиционированием.

Рассмотрим варианты:

Если расположить блоки с абсолютным позиционированием в блок с относительным, то расстояния будут уже задаваться не от края окна браузера, а от границ относительного блока.

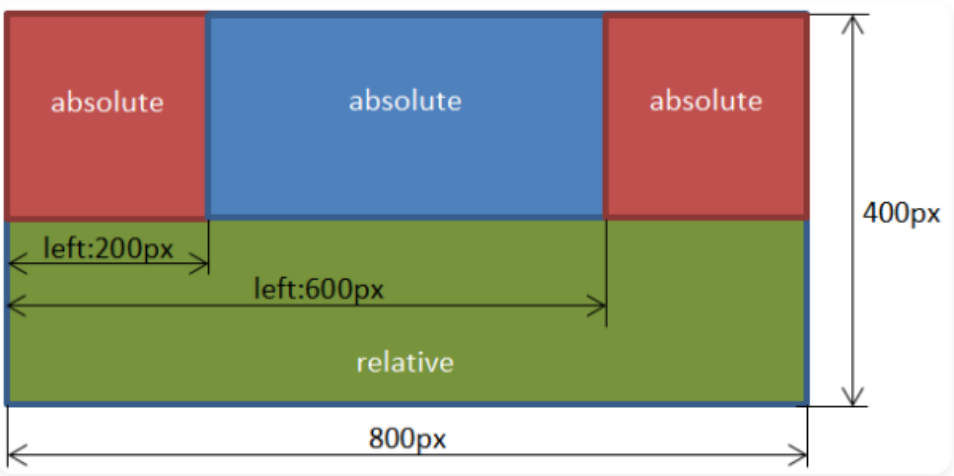
```
<body>
<!-- Зелёный блок с относительным позиционированием -->
<div style="width: 400px; height:400px; position:relative; top:50px; left:50px; border: 1px solid black; background:green;">
  <!-- Красный блок с абсолютным позиционированием -->
   <div style="width: 100px; height:100px; position:absolute; top:50px; left:50px; border: 1px solid black; background:red;"></div>
  <!-- Синий блок с абсолютным позиционированием -->
   <div style="width: 100px; height:100px; position:absolute; bottom:50px; right:50px; border: 1px solid black; background:blue;"></div>
</div>
</body>
```



2 Например: для создания фиксированных макетов состоящих из 3 блоков, выровненных

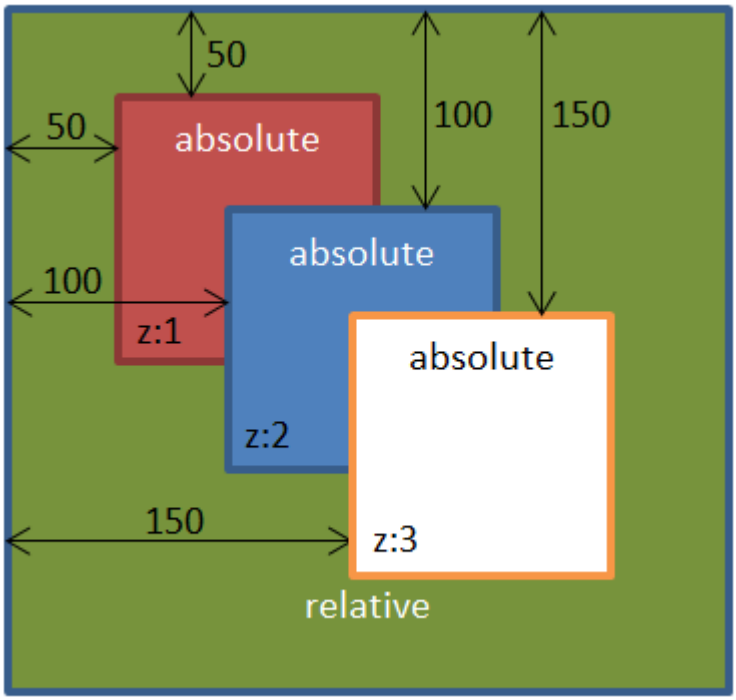
по верхнему краю. Установим высоту "400px" относительному блоку для наглядности .

```
<body>
<div style="width: 800px; height:400px; position:relative; border: 1px solid black; background:green;">
  <div style="width: 200px; height:200px; position:absolute; left:0px; border: 1px solid black; background:red;">Левый блок</div>
  <div style="width: 400px; height:200px; position:absolute; left:200px; border: 1px solid black; background:blue;">Основной блок</div>
  <div style="width: 200px; height:200px; position:absolute; left:600px; border: 1px solid black; background:red;">Правый блок</div>
</div>
</body>
```



Дополнительно к блокам можно применять свойство z-index, которое предназначено для позиционирования элементов по оси Z. Чем больше значение свойства z-index, тем ближе элемент расположен к нам, и наоборот, чем меньше значение, тем дальше расположен элемент от нас.

```
<body>
<div style="width: 300px; height:300px; position:relative; border: 1px solid black; background:green;">
  <div style="width: 100px; height:100px; position: absolute; z-index: 1; left: 50px; top: 50px; border: 1px solid black; background: red;"></div>
  <div style="width: 100px; height: 100px; position: absolute; z-index: 2; left: 100px; top: 100px; border: 1px solid black; background: blue;"></div>
  <div style="width: 100px; height: 100px; position: absolute; z-index: 3; left: 150px; top: 150px; border: 1px solid black; background: yellow;"></div>
</div>
</body>
```



Требования к оформлению отчетного материала: отчет о выполненной работе предоставлен с кодами и скриншотами веб-страниц; отчет содержит штамп
Форма контроля: отчет
Ссылки на источники: например: [2]

Практическая работа № 6
«Использование языка сценариев JavaScript при создании web-сайта»

Количество часов на выполнение: 8, из них на практическую подготовку 8
Оборудование: Windows, Notepade++, Node.Js, Chrome, Mozilla FireFox
Задание:
1 Цель работы
1.1. Изучить основы технологии JavaScript, ознакомиться с объектной моделью применительно к JavaScript
1.2. Научиться создавать скрипты. находить ошибки в сценарии и исправлять их
1.3. Научиться составлять обработки событий
2 Пояснение к работе
2.1 Краткие теоретические сведения
Гипертекстовая информационная система состоит из множества ин¬формационных узлов, множества гипертекстовых связей, определенных на этих узлах и инструментах манипулирования узлами и связями. Техно¬логия World Wide Web - это технология ведения гипертекстовых распре¬деленных систем в Internet, и, следовательно, она должна соответствовать общему определению таких систем. Это означает, что все перечисленные выше компоненты гипертекстовой системы должны быть и в Web.
Web, как гипертекстовую систему, можно рассматривать с двух то¬чек зрения. Во-первых, как совокупность отображаемых страниц, свя¬занных гипертекстовыми переходами (ссылками — контейнер ANCHOR). Во-вторых, как множество элементарных информационных объектов, составляющих отображаемые страницы (текст, графика, мобильный код и т.п.). В последнем случае множество гипертекстовых пе¬реходов страницы — это такой же информационный фрагмент, как и встроенная в текст картинка.
При втором подходе гипертекстовая сеть определяется на множест¬ве элементарных информационных объектов самими HTML-страница¬ми, которые и играют роль гипертекстовых связей. Этот подход более продуктивен с точки зрения построения

отображаемых страниц «на лету» из готовых компонентов.
При генерации страниц в Web возникает дилемма, связанная с архи¬тектурой «клиент-сервер». Страницы можно генерировать как на стороне клиента, так и на стороне сервера. В 1995 году специалисты компании Netscape создали механизм управления страницами на клиентской сторо¬не, разработав язык программирования JavaScript.
Таким образом, JavaScript — это язык управления сценариями про¬смотра гипертекстовых страниц Web на стороне клиента. Если быть более точным, то JavaScript — это не только язык программирования на стороне клиента. Liveware, прародитель JavaScript, является средством подстано¬вок на стороне сервера Netscape. Однако наибольшую популярность JavaScript обеспечило программирование на стороне клиента.
Основная идея JavaScript состоит в возможности изменения значе¬ний атрибутов HTML-контейнеров и свойств среды отображения в про¬цессе просмотра HTML-страницы пользователем. При этом перезагрузки страницы не происходит.
На практике это выражается в том, что можно, например, изменить цвет фона страницы или интегрированную в документ картинку, открыть новое окно или выдать предупреждение.
Название «JavaScript» является собственностью Netscape. Реализа¬ция языка, осуществленная разработчиками Microsoft, официально на¬зывается Jscript. Версии JScript совместимы (если быть совсем точным, то не до конца) с соответствующими версиями JavaScript, т.е. JavaScript явля¬ется подмножеством языка JScript.
JavaScript стандартизован ECMA (European Computer Manufacturers Association — Ассоциация европейских производителей компьютеров). Соответствующие стандарты носят названия ECMA-262 и ISO-16262. Этими стандартами определяется язык ECMAScript, который примерно эквивалентен JavaScript 1.1. Отметим, что не все реализации JavaScript на сегодня полностью соответствуют стандарту ECMA. В рамках данного курса мы во всех случаях будем использовать название JavaScript.
Понятие объектной модели применительно к JavaScript
Для создания механизма управления страницами на клиентской сто¬роне было предложено использовать объектную модель документа. Суть модели в том, что каждый HTML-контейнер — это объект, который ха¬рактеризуется тройкой:

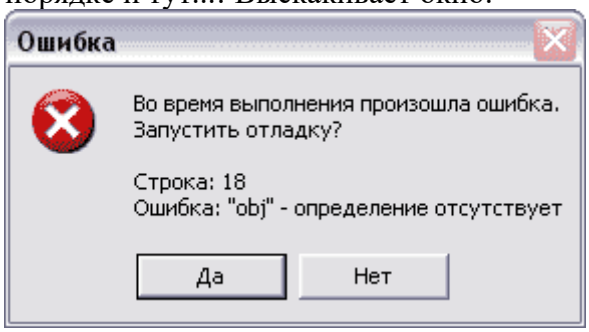
- свойства
- методы
- события

Объектную модель можно представить как способ связи между стра¬ницами и браузером. Объектная модель - это представление объектов, методов, свойств и событий, которые присутствуют и происходят в про¬граммном обеспечении браузера, в виде, удобном для работы с ними ко¬да HTML и исходного текста сценария на странице. Мы можем с ее помо¬щью сообщать наши пожелания браузеру и далее — посетителю страницы.
Браузер выполнит наши команды и соответственно изменит страницу на экране.
Объекты с одинаковым набором свойств, методов и событий объе¬диняются в классы однотипных объектов. Классы — это описания воз¬можных объектов. Сами объекты появляются только после загрузки доку¬мента браузером или как результат работы программы. Об этом нужно всегда помнить, чтобы не обратиться к объекту, которого нет.
JavaScript — это не HTML! Однако у JavaScript и HTML очень похожие правила:

- JavaScript располагается внутри документа HTML
- JavaScript сохраняется в виде текста вместе с документом HTML.

+ Главная же разница в том, что в HTML имеет довольно расплывчатые правила. Не имеет значения, сколько пробелов вы оставляете между словами или абзацами. По правде говоря, HTML можно было бы писать одной сплошной строкой. Совсем другое дело

JavaScript. У него четкая форма. И пренебрегать ею можно лишь изредка.
1. Введение
Пример скрипта:
<html>
<head>
<title>1</title>
</head>
<body>
<SCRIPT LANGUAGE="javascript">
document.write("<FONT COLOR='RED'>Это красный текст</FONT>")
</SCRIPT>
</body>
</html>
Результат
Это красный текст
Разбор скрипта:
<SCRIPT LANGUAGE="javascript">
Это код HTML, который дает браузеру понять, что с этого места начинается JavaScript. Все скрипты начинаются с такой команды. Сущесвуют и другие типы скриптов, например, VBS или LiveScript, поэтому эта команда не даст браузеру запутаться.
</SCRIPT>
...заканчивается любой JavaScript без исключений.
Далее основная часть скрипта:
document.write("<FONT COLOR='RED'>Это красный текст</FONT>")
Состав скрипта: указывается DOCUMENT (документ HTML) и те изменения, которые в нем произойдут — что-то будет написано (WRITE). То, что будет написано, находится в скобках.
DOCUMENT представляет собой object(объект). Слово WRITE (писать), отделенное точкой, называется method (методом объекта). Таким образом, скрипт попросту говорит: «Возьмите объект (что-то, уже существующее) и припишите что-то к нему». Текст в скобках называется instance (примером метода), он передает то, что происходит, когда метод воздействует на объект. Имейте в виду, что текст внутри скобок находится в кавычках. Никогда нельзя про них забывать. Текст в кавычках представляет собой простой HTML. Команда <FONT>, которая делает текст красным. Обратите внимание, что дальше идут одинарные кавычки. Если поставить двойные, JavaScript решит, что это конец строки, и получится, что только часть вашего текста будет применена к объекту, а это уже ошибка. Запомните: внутри двойных кавычек ставятся одинарные.
2. Сведения об ошибках
Если вы хоть раз пытались написать JavaScript или вставить готовый на свою страницу, тогда вам известно, что этот номер входит в программу развлечений. Вроде бы уже все в порядке и тут...! Выскакивает окно:



Сообщение об ошибке
В основном бывают ошибки двух типов: синтаксиса и сценария. Ошибка синтаксиса

означает опечатку или пропущенный текст. Ошибка сценария значит, что вы перепутали местами команды или вставили неправильные. Так или иначе, дело в одном — где-то вы напутали. Существуют программы, которые помогают исправлять ошибки, этот процесс называется «debugging» («уничтожение багов, ошибок»), но всеже лучше делать это вручную. На самом деле это даже легче, чем можно подумать.

Исправление ошибок

Говорят, что наилучший способ исправить ошибку — это ее не совершать, но сказать проще, чем сделать. Тем не менее можно свести ошибки к минимуму, пользуясь текстовым редактором без полей. Кроме того, отводите каждой команде JavaScript отдельную строку. Ни к чему разбивать длинные строки на несколько коротких. Это само по себе может привести к ошибке. И все же, готов спорить, что каждый раз, принимаясь за скрипты, вы будете получать такие сообщения. Так уж давайте разберемся, как их устранять.

В этих всплывающих окошках есть одна замечательная вещь: они сами говорят, где и в чем состоит проблема. Взгляните на сообщение. Это ошибка сценария, и находится она на строке 23. Более того, сообщение прямо говорит, в чем состоит ошибка. Разве не здорово было бы завести такой порядок и в HTML?

Строку с ошибкой нужно отсчитывать от самого верха документа HTML, а не от первой строки JavaScript. Например, в приведенном ниже документе допущена ошибка на строке 9. Это ошибка синтаксиса, так как пример (instance) не заканчивается на той же строке, где и начался. Видите, как скобка перескочила на следующую строчку?

<HTML> <HEAD> <TITLE></TITLE> </HEAD> <BODY> <SCRIPT LANGUAGE="javascript"> document.write("text for the page") </SCRIPT> </BODY> </HTML>

Но почему ошибка на строке 9, а не 8? Потому, что вы начинаете считать с верхнего края документа HTML, не пропуская ни одной строчки. Вот этот документ еще раз с пронумерованными строчками.

(строка 1) <HTML> (строка 2) <HEAD> (строка 3) <TITLE></TITLE> (строка 4) </HEAD> (строка 5) <BODY (строка 6) (строка 7) <Script Language="JavaScript"> (строка 8) document.write("text for the page" (строка 9)) (строка 10) </SCRIPT> (строка 11) </BODY> (строка 11) </HTML>

Так что, считайте все строки, даже пустые.

Как только вы нашли строку с ошибкой, необходимо решить, что делать. Чаще всего это будет синтаксическая ошибка — либо разорванная строка, либо опечатка, либо двойные кавычки вместо одинарных и тому подобное. Если это ошибка сценария, значит, команда, на которую указывает сообщение, не укладывается в логическую последовательность. Например, команда вызывает кнопку, а в скрипте на самом деле указано текстовое поле.

+ Многократные сообщения JavaScript — это чрезвычайно логичный язык, требующий, чтобы все шло своим чередом, друг за дружкой. Допустим, у вас 10 ошибок в длинном скрипте. Сообщения накладываются одно на другое, и последняя обнаруженная компьютером ошибка окажется сверху. Не набрасывайтесь на нее сразу, возможно, в действительности ее даже не существует. Может случиться так, что первая ошибка в скрипте и вызовет все остальные. Так что исправлять их следует от начала документа HTML.

. Дата и время

Нам известно, что существует такой объект как документ. Иначе в нем ничего нельзя было бы написать. Существуют семь методов объекта: getDay(), getDate(), getMonth(), getYear(), getHour(), getMinute(), и getSecond() (получитьДень, Число, Месяц, Год, Час, Минуту, Секунду). Все они уже существуют, их можно взять и поместить на страницу. Проблема в том, что это всего лишь методы. Для воздействия им нужен объект, а документ для этих целей не годится... так что придется самим придумывать новый объект.

Ошибка 2000!

С 1 января 2000 вместо двух последних цифр года, как раньше, getYear в Експлорере показывает полный четырехзначный номер, а в Нескейпе — число 100. Может быть, Нетскейп заделает эту дырку, а до тех пор лучше пользоваться командой getFullYear, которую, кажется, правильно воспринимает и тот, и другой. Правда, старая «Опера» ее вообще не воспринимает.

Скрипт

```
<Script Language="JavaScript">
Now = new Date();
document.write("Сегодня " + Now.getDate()+ "-" + Now.getMonth() + "-" + Now.getFullYear() + ". Вы зашли на мою страницу ровно в: " + Now.getHours() + ":" + Now.getMinutes() + " и " + Now.getSeconds() + " секунд.")
</SCRIPT>
```

Кстати, строка document.write не должна прерываться.

Результат

Сегодня 14-11-2009. Вы зашли на мою страницу ровно в: 15:34 и 23 секунд.

Разбор скрипта

Методы Дата и Время

Посмотрите на скрипт. Видите, скрипту дается команда поместить в документ число, месяц, год, часы, минуты и секунды. Несколько дополнительных слов объясняют, на что вы смотрите. Все это было создано с помощью метода getЧто-либо(). Обратите внимание на заглавную букву. Сначала «g» в нижнем регистре, потом Заглавная буква.

Во-первых, помните, что все это цифры. Даже метод getDay(), который отвечает за день недели, выражается цифрой от единицы до семи. Начнем с месяца. Как уже говорилось раньше, getMonth() — это метод, отвечающий за месяц. Теперь задумаемся о том, на какой объект воздействует метод getMonth().Метод (method) воздействует на объект (object).

Может показаться, что getЧто-либо() — это метод объекта document. Вовсе нет. Метод документа — write. getMonth() на самом деле является методом объекта Date. Взгляните на скрипт еще раз. Date занимает отдельное место в команде:

```
Now = new Date();
```

Устанавливаем объект, с которым будет работать метод getMonth(). Имея дело с датой и временем, всякий раз пользуйтесь той же схемой. Нужно создать объект. Наш объект называется Now (сейчас). Название не имеет значения, если объект получил оригинальное имя, которое больше нигде в JavaScript не встречается. Команда говорит: Now — это объект, который представляет new Date() (новую Дату). Дата обязательно должна быть новой. Таким способом вы будете получать новую дату каждый раз, когда заходите на страницу или обновляете ее.

Обратите внимание и на точку с запятой в конце. Она указывает на то, что строка JavaScript закончена. Без нее браузер решил бы, что команда продолжается на следующей строке. Ошибка.

У нас есть объект, на который может воздействовать метод getMonth(). Нам нужно напечатать месяц на странице, значит, где-то должна быть команда document.write(). Нам также известно, что текст в скобках будет виден на странице, так давайте напишем все это, следуя логике: Сначала пишем <Script Language="JavaScript">. Прежде чем приступить к getMonth(), необходимо создать объект. Убедитесь, что строка заканчивается точкой с запятой. Теперь можно вставлять утверждение document.write. Текст в скобках после document.write оформляем по правилам. Текст, видимый на странице, должен быть окружен двойными кавычками (одинарные кавычки для кода HTML внутри двойных).

Новое правило: сочетание текста и команд требует знака «плюс» + между элементами. Объект и метод разделены точкой, так что команда напечатать месяц выглядит так: Now.getMonth().

Новое правило: Now.getMonth() — это не текст, который должен быть виден на странице, а команда, которая указывает месяц. Поэтому не нужно ставить ее ни в какие кавычки.

Заканчиваем командой </SCRIPT>.
И вот что у нас получилось: <Script Language="JavaScript">
//Скрипт напечатает на странице номер месяца
Now = new Date(); document.write("Сейчас месяц " + Now.getMonth())
</Script>
Посмотрите на первый скрипт еще раз. Длинная строка уже не кажется такой страшной. Это просто объект Now и метод getЧто-либо() после него. Элементы даты разделены дефисами. Помните, дефис должен быть виден на странице, поэтому его следует ставить в кавычки. Все части связаны между собой плюсами +.
Пробелы
Сколько бы пробелов вы не ставили до и после знаков плюс, это никак не повлияет на видимый результат. Элементы пойдут сплошным текстом. Поэтому, если вам нужны пробелы, добавляйте их в части текста в кавычках. Например:
"Сейчас ровно "
Когда скрипт отпечатается на странице, пробел окажется на своем месте. Помните: это не HTML. В Javascript существуют свои правила относительно пробелов.
Длинная строка
Не будем разбирать всю строку, оставим только строку с датой. Вот как это выглядит:
document.write("Сегодня " + Now.getDate() + "-" + Now.getMonth() + "-" + Now.getFullYear() + ". Вы зашли на страницу ровно в: " + Now.getHours() + ":" + Now.getMinutes() + " и " + Now.getSeconds() + " секунд.")
Начинаем с «Сегодня», прибавив пробел в конце. Следом знак плюс. Now.getDate() без кавычек, потому что нам нужен не текст, а цифры. Еще плюс. Потом дефис в кавычках, чтобы отделить следующие цифры. Никаких пробелов, потому что они должны стоять вплотную. Плюс. Потом Now.getMonth без кавычек, чтобы у нас был месяц. Плюс. Еще дефис в кавычках, чтобы он был виден на странице. Плюс. Еще один метод Now.getFullYear сообщит год. Продолжайте дальше по этой схеме, и скрипт напечатает именно то, что вы хотите. Тогда вы всем сможете сказать, который час.
Кое-что об одной из самых интересных заморочек JavaScript. Должно быть, вы заметили, что номер месяца на один меньше, чем нужно. Почему? Цифры сообщает нам JavaScript, а JavaScript любит считать от нуля. То есть, январь нулевой месяц и так далее.
Вы присваиваете new Date() имя, как уже делали раньше. Затем присваиваете имя команде, которая вызывает месяц. Ниже я назвал ее mpo (Месяц Плюс Один). И прибавляете к ней единицу.
<Script Language="JavaScript"> Now = new Date(); var mpo = Now.getMonth(); var mpo1 = mpo + 1 document.write("Сейчас месяц " + mpo1 + "."); </SCRIPT>
И вот что у вас получилось:
Сегодня месяц 12.
4. Команда onMouseOver
Теперь рассмотрим события (events). События (event) и обработчики событий (event handler) относятся к JavaScript, но они скорее «встроены» в HTML-код, а не существуют самостоятельно, как те скрипты, которые были рассмотрены до этого. Они входят в структуру документа HTML, не требуя команд <SCRIPT> и </SCRIPT>. Сами они не скрипты, а скорее область взаимодействия между вашей страницей и читателем. События — это то, что происходит. Они добавят динамики вашему сайту. Увидев их, посетители скажут: «Ух ты!», а сочинять длинные скрипты совсем не нужно. Среди разнообразных обработчиков событий для начала мы рассмотрим один, самый популярный, — onMouseOver (навести мышь).
Скрипт
<A HREF="http://www.narod.ru" onMouseOver="window.status='Бесплатный хостинг'; return true">Ссылка</A> Все это должно быть на одной строке.
Эффект

Ссылка

Наведите курсор на ссылку и посмотрите на строку состояния в окне браузера.

Разбор скрипта

Вы уже знаете достаточно, чтобы понять смысл написанного. Давайте быстро разберем скрипт и попробуем его изменить. Во-первых, onMouseOver (обратите внимание на заглавные буквы) представляет собой обработчик событий (Event Handler) гипертекстовой ссылки. Он используется внутри гиперссылки. Формат ссылки остается без изменений. Те же команды и те же двойные кавычки. onMouseOver ставится сразу же после адреса URL.

Событие (Event) приводится в действие, когда браузер распознает onMouseOver="". Схема уже должна казаться вам немного знакомой: два элемента, разделенных точкой. До сих пор это означало, что один является объектом, а другой методом. Но не в этом случае. Объектом является window (окно), оно существует; status (статус) представляет собой property (свойство) окна. Это небольшой участок окна, где должен разместиться следующий текст. Это проще запомнить, если представить, что метод обычно выражается глаголом, как write (писать) или get (получить). Свойство выражается существительным и существует как небольшая часть элемента, стоящего перед точкой. Если у window есть изменяемое свойство status, значит, можно изменить и другие свойства окна.

После window.status следует знак равенства = и то, что должно произойти. В данном случае это текст в одинарных кавычках. Он появится в строке состояния, когда вы наведете курсор на ссылку. Пожалуйста, обратите внимание на точку с запятой в конце строки.

return true Эти два слова имеют не последнее значение. Они дают скрипту указание проверить, есть ли строка состояния. Если отчет (return) соответствует действительности (true), тогда происходит событие. Обратите внимание, что текст в строке состояния уже не изменяется и не изменится, сколько раз вы не наводили бы на нее курсор.

Поэтому не нужно злоупотреблять такими трюками — человек, пришедший к вам на сайт, имеет право знать точный адрес каждой ссылки и другую информацию о загрузке страницы, которую выдает строка состояния.

Другие свойства

Вернемся к свойствам. Если они есть у окна, другие объекты тоже должны иметь свойства. Цвет фона – это свойство. В HTML цветом фона страницы управляет команда BGCOLOR. То же самое и здесь, только обязательно соблюдайте регистр. В JavaScript он пишется bgColor. Подумаем, как создать ссылку. которая изменяла бы фон страницы с помощью обработчика onMouseOver.

Во-первых, раз это ссылка, то сохраним ту же схему.. Заменим window на document. Для изменения цвета фона объекта, заменим status на bgColor. Необходимо, чтобы новый цвет оставался независимо от того, сколько раз мы будем наводить курсор на ссылку, потому вставляем return true после точки с запятой.

Вот что у нас получилось...

Не щелкать

Для того, чтобы два события произошли одновременно, нет необходимости разделять команды точкой с запятой, так как это означает конец.

+ Новое правило: ставьте запятую, чтобы отделить друг от друга разные команды JavaScript, происходящие одновременно. Помните, в кавычки ставятся отдельные элементы вроде текста. Раз нам нужно, чтобы обе команды действовали одновременно, ставим кавычки в самом начале первой и в самом конце второй. Таким образом мы показываем браузеру, что все это одно событие. Однако нам еще могут понадобиться одинарные кавычки...

<a href="http://www.narod.ru" onMouseOver="document.bgColor='334775',
onMouseOver=window.status='Бесплатная почта'; return true">Не щелкать

. Обработчики событий

Сейчас вы уже представляете себе, что такое некоторые события и команды, которые ими управляют. Давайте рассмотрим, как действуют другие. Все они работают по одной схеме. Так что если вам не чужда логика, вы легко сможете поместить их на свои страницы.

Команды и эффекты

Команда onClick (на щелчок)

onMouseOver запускает событие, если навести курсор на ссылку. Следовательно, щелкнув по ссылке, можно с таким же успехом запустить событие через onClick. Чтобы продемонстрировать действие команды, воспользуемся методом alert. Если вы сделали прошлое задание, то знаете, что это такое. Вот еще раз его схема:

alert('текст, который появится в окне')

Таким образом, получаем:

 Жмите сюда

И вот что это нам дает (когда вы нажмете на ссылку, она сработает):

Помните, что внутри одинарных кавычек нельзя употреблять слова с апострофами ', иначе браузер поймет их, как окончание текста, а это не входит в ваши намерения. Ошибка.

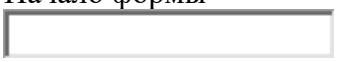
Команда onFocus (на фокус)

Это замечательная команда, которая вызывает действие, когда пользователь «фокусируется» на элементе страницы. Это годится для форм: флажков (checkbox) и текстовых полей (textbox). Вот пример:

<form> <input type="text" size=30 onFocus="window.status='Текст в строке состояния';"> </form>

Вот что вы получаете (щелкните в поле ввода и посмотрите на строку состояния):

Начало формы

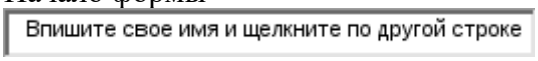


Конец формы

Команда onBlur (на потерю фокуса)

Если можно сосредоточиться на объекте, значит, можно и «потерять фокус». onBlur позволяет сообщить пользователю о том, что он изменил свой ответ. Этот обработчик не так часто используется, но вот вам пример. Внизу у меня строка для ввода текста, в которой уже что-то написано. Измените текст и уведите курсор, как если бы вы перешли к следующему предмету в списке.

Начало формы



Конец формы

<form> <input type="text" size=45 value="Впишите свое имя и щелкните по другой строке" onBlur="alert('Вы изменили ответ — уверены, что он правильный?');"> </form>

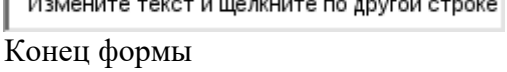
Команда onChange (на изменение)

Действие этой команды очень похоже на действие предыдущей, onBlur. Ее главная задача — проверка. Этот обработчик события проверяет, сделал ли пользователь то, что вы от него просили. Пример очень похож на предыдущий, но действует все-таки по-другому.

<form> <input TYPE="text" size=45 value="Измените текст и щелкните по другой строке" onChange="window.status='Текст был изменен';"> </form>

Это дает вам следующее...

Начало формы



Конец формы

Команда onSelect (на выделение)

Эта команда работает так же, как и три предыдущие, отмечая, что в поле ввода произошли изменения, — в данном случае был выделен текст.

Команда onSubmit (на отправку)

Это очень популярная команда. Она позволяет вызвать какое-либо действие, когда вы

нажимаете кнопку Submit (отослать, отправить). Многим очень хочется, чтобы после того, как пользователь нажимает на кнопку, у него на экране появлялась страница с надписью: «Спасибо, что вы нам написали».
Формат такой:
<form> <input TYPE="submit" onSubmit="parent.location='thanksalot.html'";> </form>
Вот что у вас выходит (щелкайте по кнопке):
Начало формы
Конец формы
Поглядите, у нас новая команда. parent.location — это стандартная схема ссылки на другую страницу. Можно подумать, что parent (источник) — это объект, а location (местонахождение) — метод. Неверно. В данном случае parent является свойством окна браузера, а location — объектом, который появится в этом окне. То есть для ясности просто имейте в виду, что parent.location=" означает ссылку.
+ Команды onLoad и onUnload (на вход и выход)
Перечень используемого оборудования
2.2.1 Персональный компьютер
2.2.2 Описание практической работы
3 Задание
3.1 Напишите скрипт так, чтобы вышли две строки текста, красная и синяя. Необходимо дописать несколько команд Javascript, а не просто добавить немного HTML к приведенному примеру. На странице должно оказаться следующее:
Иван рубил дрова, Варвара топила печь.
2.Скопируйте скрипт на страницу и сохраните ее.
<HTML> <HEAD> <TITLE></TITLE> </HEAD> <BODY>
<SCRIPT language=JavaScript>
..t
dothis = new Date()
month = dothis.getMonth()
month = (month * 1) + 1
day = dothis.getDate()
year = dothis.getFullYear()
document.wrte(" ",day,"/",month,"/",year," ")
</SCRIPT>
</BODY> </HTML>
Загружая страницу, браузер должен выдать вам два сообщения об ошибке. Исправьте их.
Если он заработает, то на странице появится сегодняшняя дата. (Подсказка: возможно, сначала вы получите только одно сообщение. Вторая ошибка появится, когда вы исправите первую.)
2.Напишите скрипт, который поместит на вашу страницу дату, разделенную дробями /. Приветственный текст должен быть зеленого цвета. Также отметьте, что это вы написали скрипт.
3.Новый метод, alert() (предупредить) - он вызывает небольшое диалоговое окно с текстом и кнопкой OK. Попробуйте сделать так, чтобы окно предупреждения всплывало при наведении курсора на ссылку. Вот формат команды:
alert('текст в окошке')
Подумайте хорошенько, решите, что должно произойти сначала, что потом.
2.Создайте форму, которая будет взаимодействовать с пользователем. Форма должна иметь три элемента: поле ввода с просьбой ввести имя; два поля для флажков с вопросом о том, что больше нравится пользователю, мороженое или шоколад; кнопку отправки данных. С каждым элементом должно произойти следующее: При вводе имени в строке состояния должны появиться слова: «Впишите сюда свое имя». Два поля с флажками должны отослать в строку состояния слова: «Вы выбрали...» и выбор пользователя. При

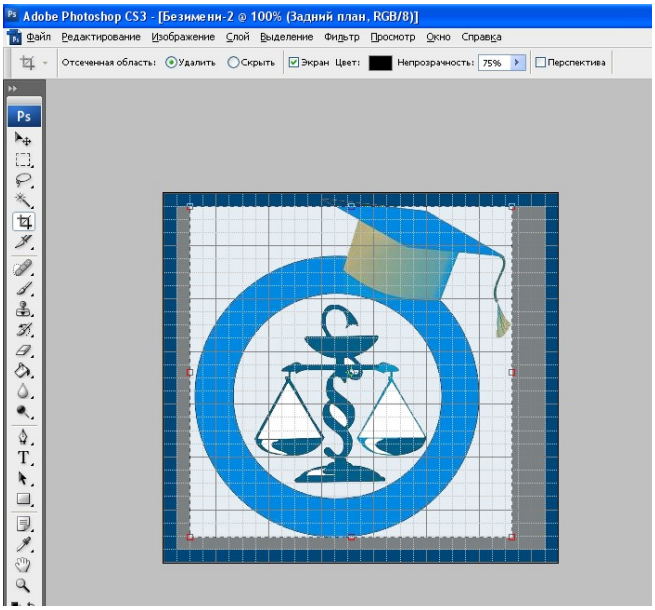
нажатии на кнопку должно выскочить окно предупреждения, благодарящее пользователя за участие в опросе.
4 Контрольные вопросы
4.1 Что позволяет вызывать команда onSubmit?
4.2 В чем суть объектой модели JavaScript?
4.3 Дайте определение понятию JavaScript?
Требования к оформлению отчетного материала:
5 Содержание отчета
5.1 Наименование работы
5.2 Цель работы
5.3 Задание
5.4 Выводы по работе
5.5 Ответы на контрольные вопросы
Требования к оформлению отчетного материала: отчет о выполненной работе предоставлен с кодами и скриншотами веб-страниц; отчет содержит штамп
Форма контроля: отчет
Ссылки на источники: например: [2]

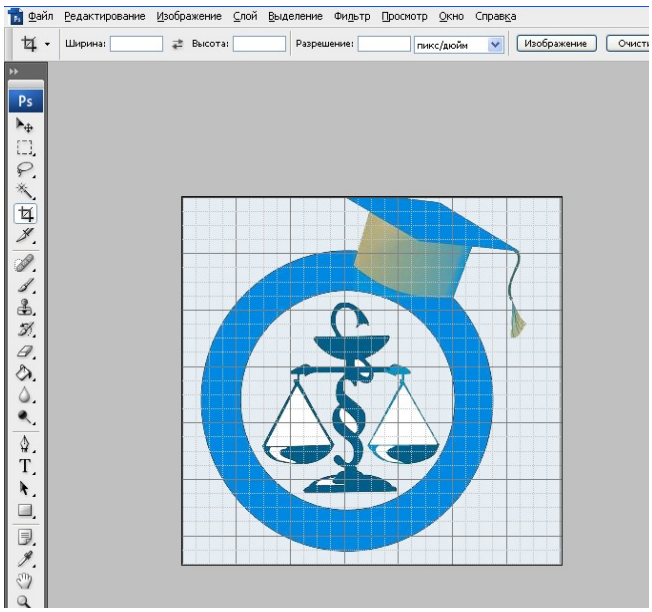
Практическая работа № 7

«Подготовка и оптимизация графики на web-странице»

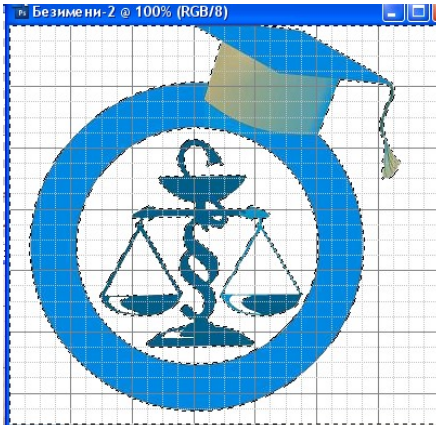
Количество часов на выполнение:2, из них на практическую подготовку 2
Оборудование: Adobe Photoshop CS5
Задание:
Обрезка изображения. Удаление фона

1. Запустите программу Adobe Photoshop CS5.
2. Откройте файл Логотип.gif из папки “Заготовки”.
3. Необходимо обрезать рамку и удалить фон.
4. Выберите инструмент «Рамка»  . Обрежьте ненужную часть изображения.



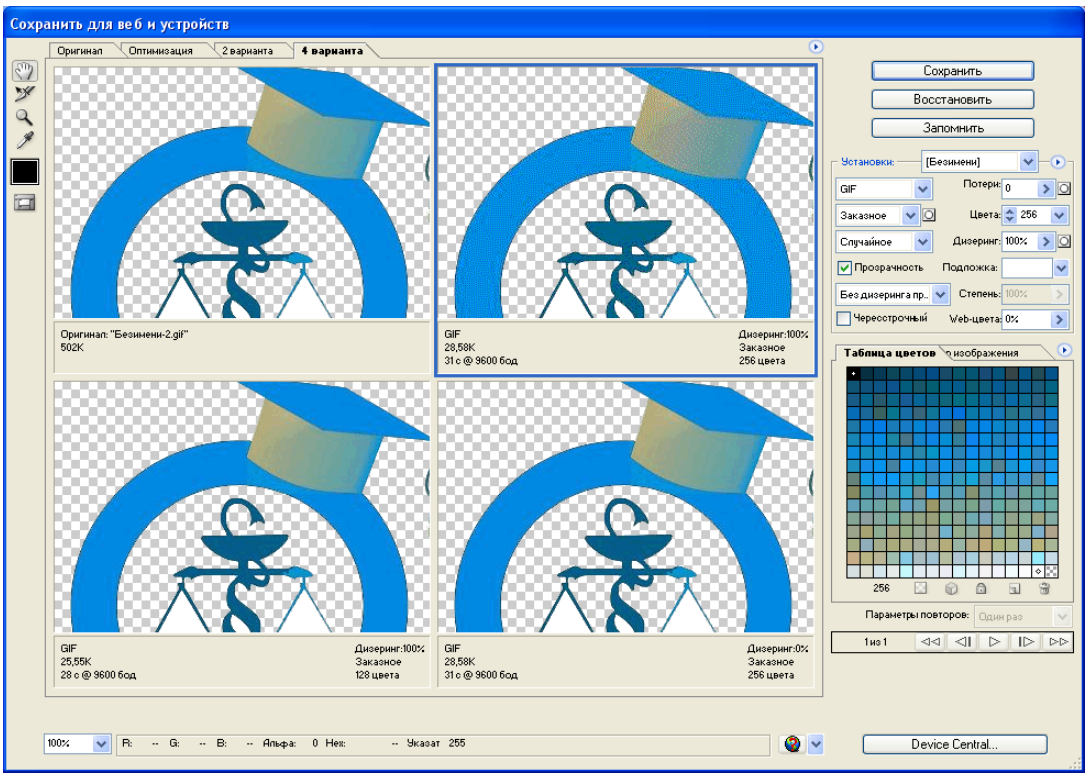


5. Выберите инструмент «Волшебная палочка»  .
6. Щелкните указателем мыши в любом месте фона. В результате логотип должен выделиться пунктирной линией. Нажмите клавишу Delete, фон будет удален.
7. Нажмите комбинацию клавиш CTRL+D, чтобы снять выделение.

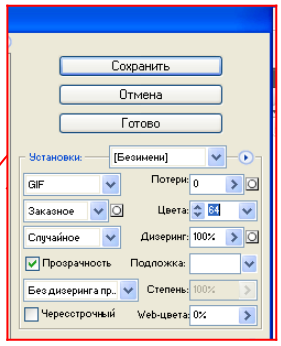


Оптимизация формата gif.

1. Выберите команду «Файл» - «Сохранить для Web и устройств…»

2. 

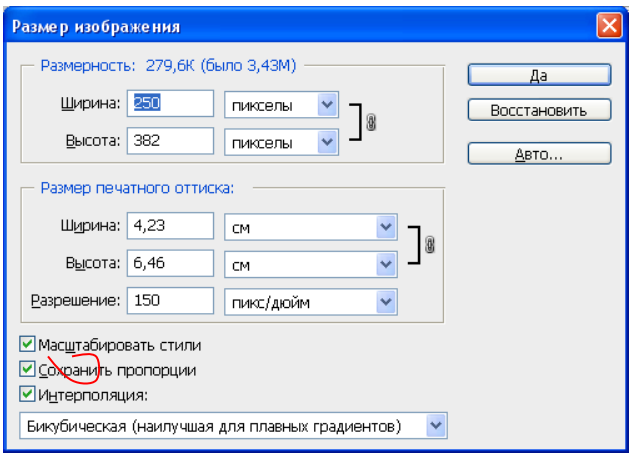
3. В диалоговом окне выберите закладку 4 варианта.
4. 2 окно - выберите Формат - gif. Установите прозрачность.
5. 3 окно - выберите Формат - jpeg.
6. 4 окно - выберите Формат – png 24.
7. Сравните три варианта документа. Выберите вариант с наименьшим размером файла.
8. Сохраните документ.

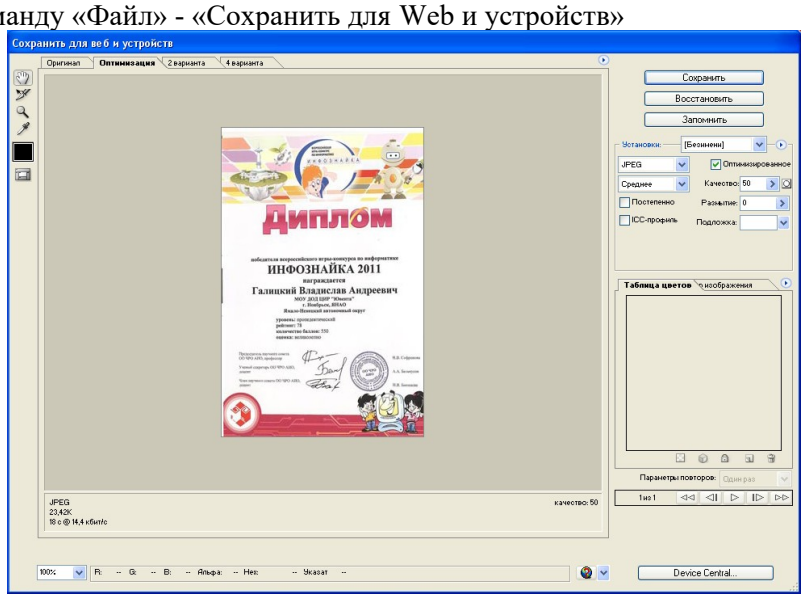


Оптимизация формата jpeg

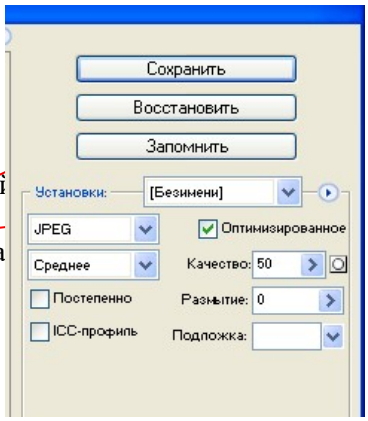
1. Откройте в программе Adobe Photoshop CS5 файл Диплом.jpeg из папки «Заготовки».
2. *Исходные данные изображения: 29,72 см х 21,59 см, размер 6.4 МГб. Это очень большой размер изображения и большой объем файла для web-страницы и его необходимо уменьшить к качества.*
3. Вырезать фрагмент докум

4. *Для исправления Перспек необходимо включить фл*
5. Выберите пункты меню «И
6. Убедитесь, что флажок «Со

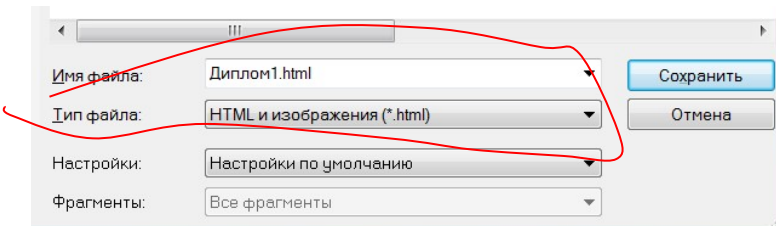
7. 

8. В поле Ширина поставьте значение 250. В поле Высота значение измениться пропорционально. Нажмите кнопку ОК.
9. Установите масштаб изображения равным 100%.
10. Выберите команду «Файл» - «Сохранить для Web и устройств»
11. 

12. Переключитесь на вкладку Оптимизация и вы увидите, что размер файла стал приблизительно 29 Кб.
13. В палитре Оптимизация настройте установки:
14. Формат - jpeg
15. Качество - 50.



16. Посмотрите на размер вашего файла, он стал примерно 23 Кбай
17. Сохраните изображение.
18. В диалоговом окне сохранения файла введите имя файла и в ра файла должно стоять Только изображение.



Требования к оформлению отчетного материала: отчет о выполненной работе предоставлен с кодами и скриншотами веб-страниц; отчет содержит штамп
Форма контроля: отчет

Ссылки на источники: [2]

Практическая работа № 8
«Создание баннера для web-страницы»

Количество часов на выполнение:4, из них на практическую подготовку 4
Оборудование:
Задание:
1 Цель работы
1.1 научиться создавать баннеры для web-страниц
2 Пояснение к работе
2.1 Краткие теоретические сведения
Баннер – это рекламный блок, при щелчке по которому посетитель переходит на рекламируемый сайт. Баннеры размещают на web-страницах, для привлечения посетителей (потенциальных клиентов) на свой сайт или для продвижения бренда. В настоящее время баннеры очень активно используются в Интернете, их по праву можно считать одним из главных инструментов для проведения эффективной рекламной кампании.
Виды баннеров
Самый первый баннер появился в сети в 1994 году. За свою недолгую историю у баннеров появилось множество разновидностей и технологий на которых они основывались. Ниже приведем несколько основных видов баннеров:
- JPG баннер – один из первых разновидностей баннера появившихся в Интернете, представляет из себя статическое изображение. На данный момент считается устаревшим и не может соперничать со своими более современными собратьями.
- GIF баннер – следующая ступень эволюции баннеров. Представляет собой анимированный графический элемент созданный из набора статичных изображений (кадров). Обычно состоит из 3-5 кадров.
- Flash баннер – самая современная технология. Позволяющая создавать самые эффектные анимированные баннеры. Предоставляет обширные возможности для реализации дизайнерских идей.
- Пиксельные баннеры – обособленный вид баннеров. Если при создании JPG, GIF и Flash баннеров в большинстве случаев используются фотоизображения, что негативно сказывается на весе, пиксельные баннеры создаются путем прорисовки дизайнером каждого пикселя (пиксель – мельчайшая точка или элемент изображения) вручную. Такая технология, позволяет создать красочную анимацию при минимальном весе баннера.
- Форматы баннеров
Формат – параметры ширины и высоты баннера, измеряемые в пикселях. Например, баннер шириной 468 пикселей и высотой в 60 пикселей, обозначается как 468x60. В техническом плане ограничений в формате баннера не существует. Существуют лишь общепринятые стандарты, на которые опирается большинство владельцев web-сайтов, при проектировании рекламных мест на своем ресурсе. Вы вполне можете отойти от общепринятых форматов, если владелец рекламного места предоставляет возможность размещения «нестандартных» баннеров.
2.2 Перечень используемого оборудования
2.2.1 Персональный компьютер
2.2.2 Описание практической работы
3 Задание
3.1 Создайте 2 баннера для web-страниц на выбор, а затем их сохраните для web
Варианты заданий

Вариант 1	Вариант 2

Вариант 3

Вариант 4

Вариант 5

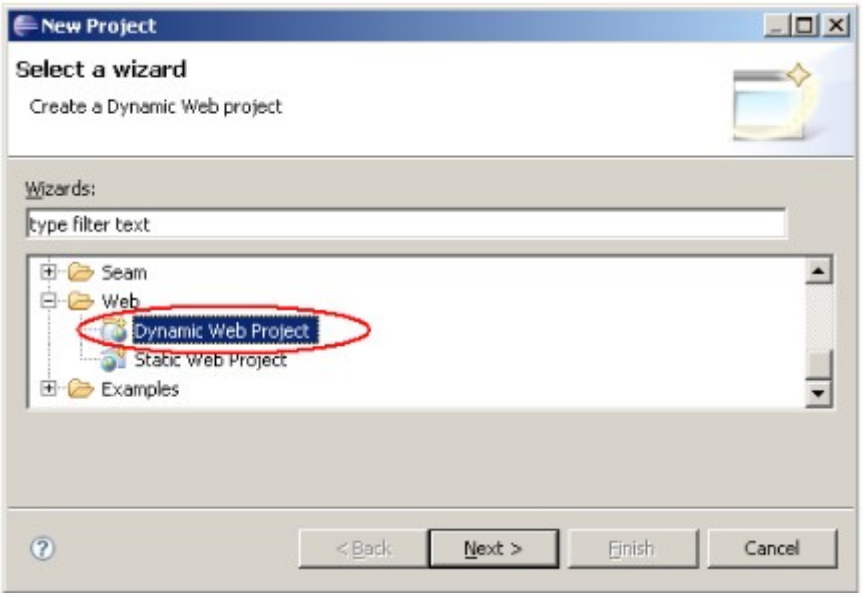
Вариант 6

4 Контрольные вопросы
4.1 Какие эффекты вы применили при изготовлении баннера?
4.2 Какие основные виды баннеров вы знаете?
4.3 Что такое баннер?
Требования к оформлению отчетного материала:
5 Содержание отчета
5.1 Наименование работы
5.2 Цель работы
5.3 Задание
5.4 Выводы по работе
5.5 Ответы на контрольные вопросы
Требования к оформлению отчетного материала: отчет о выполненной работе предоставлен с кодами и скриншотами веб-страниц; отчет содержит штамп
Форма контроля: отчет
Ссылки на источники: [2]

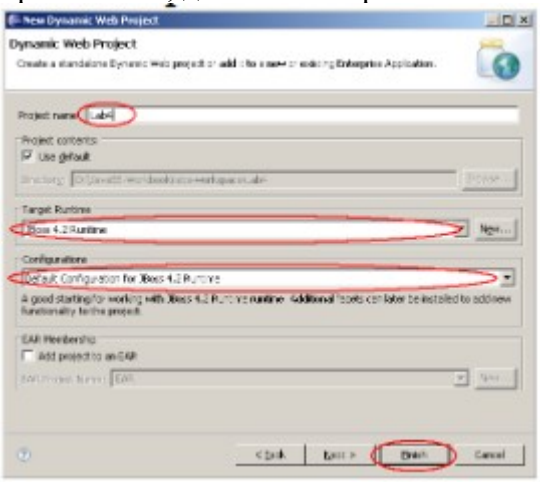
Практическая работа № 9
«Разработка эскизов веб-приложения»

Количество часов на выполнение: 4, из них на практическую подготовку 4
Оборудование:
Задание:
Цель работы:
- необходимо разработать веб-приложение, использующее сервлет для поиска нформации

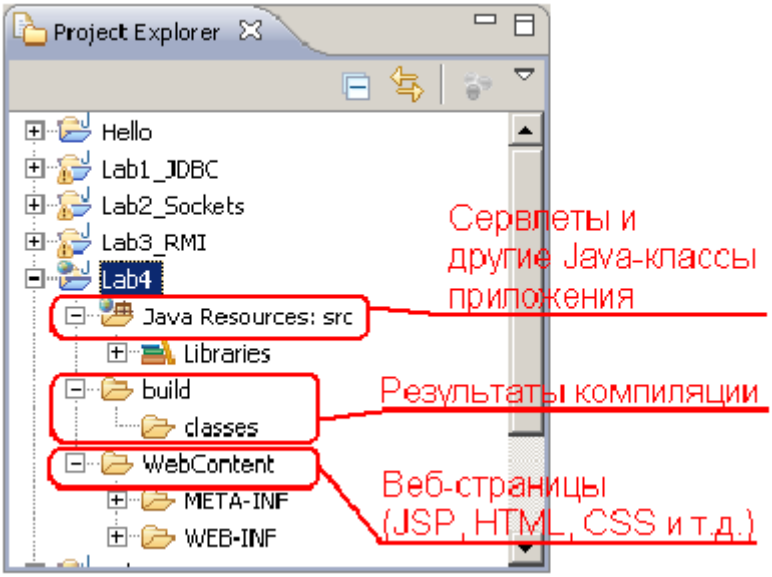
о сотрудниках организации;
- данные о сотрудниках хранятся в таблице Employee;
- для осуществления поиска пользователь указывает фамилию сотрудника и рассматривает информацию о найденных сотрудниках (возможно существование нескольких сотрудников с одинаковыми
фамилиями).
Оборудование и ПО: ПК, ОС Windows 7
Форма работы: фронтальная.
Ход работы.
Задания:
1) Выберите пункт меню File/New/Project, в окне выбора типа проекта укажите Web/Dynamic Web Project и нажмите Next.



2) Укажите имя проекта Lab4. Остальные настройки, связанные с сервером приложений, на котором будет разворачиваться веб-приложение, должны отобразиться автоматически. Нажмите Finish.



3) В результате будет создан проект следующей структуры:



Порядок и методика выполнения заданий:
Отчет: Отчет должен содержать:
– наименование работы;
– цель работы;
– задание;
– последовательность выполнения работы;
– ответы на контрольные вопросы;
– вывод о проделанной работе.
Форма контроля: отчет
Ссылки на источники: [2]

Практическая работа № 10

«Разработка прототипа дизайна веб-приложения»

Количество часов на выполнение: 4, из них на практическую подготовку 4
Оборудование:
Задание:
Цель работы:
освоить создание динамических прототипов в среде Flash, опробовать процесс тестирования прототипов и их итерационного улучшения.
Оборудование и ПО: ПК, ОС Windows 7
Форма работы: фронтальная.
Ход работы.
Задания:
1. Выбрать интерфейс материального продукта (DVD-плеера, стиральной машины и т.п.) или веб-сайта
2. С учѐтом результатов анализа конкурентов, сильных и слабых сторон интерфейса спроектировать новую, улучшенную версию интерфейса для выбранного продукта или веб-сайта.
3. Реализовать динамический прототип создаваемого интерфейса с использованием средств Adobe Flash.
4. На основе анализа задач пользователя, разработать задания для мини-тестирования прототипа с пользователем (представителем будущих целевых пользователей). В качестве пользователя, участвующего в тестировании, может выступать один из студентов вашей учебной группы.
5. На основе результатов, полученных в ходе тестирования прототипа с пользователем, спроектировать и реализовать с использованием Adobe Flash новую, ещѐ более улучшенную версию прототипа интерфейса

Порядок и методика выполнения заданий:
Отчет: Отчет должен содержать:
– наименование работы;
– цель работы;
– задание;
– последовательность выполнения работы;
– ответы на контрольные вопросы;
– вывод о проделанной работе.
Сделать вывод.
Форма контроля: отчет
Ссылки на источники: [2]

Практическая работа № 11
«Разработка схемы интерфейса веб-приложения»

Количество часов на выполнение: 4, из них на практическую подготовку 4
Оборудование:
Задание:
Цель работы:
получить практические навыки разработки пользовательских интерфейсов на этапе проектирования, включая определение цели и исходных требований к программе, анализ пользователей и создание сценариев поведения пользователей.
Оборудование и ПО: ПК, ОС Windows 7
Форма работы: фронтальная.
Ход работы.
Задания:
1. Определить предметную область и сферу применения программного продукта.
2. Определить целевую аудиторию.
3. Построить описательную модель пользователя (профиль). При необходимости — выделить группы пользователей.
4. Сформировать множество сценариев поведения пользователей на основании составленной модели.
5. Выделить функциональные блоки приложения и схему навигации между ними (структуру диалога).
Порядок и методика выполнения заданий:
Отчет: Отчет должен содержать:
– наименование работы;
– цель работы;
– задание;
– последовательность выполнения работы;
– ответы на контрольные вопросы;
– вывод о проделанной работе.
Форма контроля: отчет
Ссылки на источники: [2]