

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

**Методические указания
по выполнению практических работ**
по дисциплине «Информационные технологии в
профессиональной деятельности»

для студентов направления подготовки
09.03.02 «Информационные системы и технологии»
Профиль «Прикладное программирование в
интеллектуальных информационных системах»

Ставрополь, 2026

Введение

Целью изучения дисциплины «Информационные технологии в профессиональной деятельности» является изучение таких вопросов и понятий как информация, содержание информационной технологии как составной части информатики, эволюция информационных технологий. Изучаются базовые информационные процессы, их характеристика, базовые информационные технологии в профессиональной деятельности, прикладные информационные технологии в профессиональной деятельности, стандарты пользовательского интерфейса и др.

Целью изучения дисциплины «Информационные технологии в профессиональной деятельности» является изучение таких вопросов и понятий как информация, содержание информационной технологии как составной части информатики, эволюция информационных технологий. Изучаются современные информационные процессы, их характеристика, информационные технологии и программирование в профессиональной деятельности, современные прикладные информационные технологии в профессиональной деятельности, стандарты пользовательского интерфейса.

ПРАКТИЧЕСКАЯ РАБОТА №1. ВВЕДЕНИЕ В ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ. ОЦЕНКА КАЧЕСТВА ИНТЕРНЕТ РЕСУРСА

Цель работы: научиться оценивать по необходимым характеристикам качество Интернет ресурса.

1. Изучить лекционный материал

Существуют четыре основные причины, по которым пользователи на одни сайты возвращаются, а на другие - нет. Эти четыре фактора - основа хорошего веб-дизайна, поскольку именно этого больше всего хотят пользователи. Они могут быть собраны воедино с помощью аббревиатуры HOME:

- Высокое качество содержания (High-quality content)
- Частые обновления (Often update)
- Минимальное время загрузки (Minimal download time)
- Простота использования (Ease of use)

Чтобы перейти от HOME-дизайна к дизайну "HOME-RUN", добавьте три дополнительных качества:

- Соответствие потребностям пользователя (Relevant to user`s needs)
- Уникальность в Интернете (Unique to the online medium)
- Ориентированная на Интернет корпоративная культура (Net-centric corporate culture)

Концепция высокого качества веб-страниц общего характера сформулировать:

- красивая форма дизайна, графика;
- умное использование HTML кода и подключаемых модулей, которые делают сайт интерактивным и удобным в пользовании;
- наличие инновационных материалов;
- наглядность, интуитивность интерфейса

Судить о качестве Интернет ресурса и его предназначенности для своих профессиональных целей следует по нижеследующим признакам.

URL домены

Каждый унифицированный указатель информационного ресурса (URL) заканчивается суффиксом, который указывает на происхождение и назначение сайта:

- *.com Коммерция
- *.org Международная организация
- *.gov Государственный департамент
- *.edu Образование
- *.mil Военное дело
- и т.д.

Например:

- whitehouse.gov- сайт правительства США, президентский сайт
- whitehouse.org - пародия на государственный сайт
- whitehouse.com - порно сайт

Авторы

Сведения об авторе, ответственном лице, адрес e-mail и другие сведения об "ответственности" обычно располагаются внизу страницы. Следует иметь в виду, что даже страницы солидных профессионалов могут иметь субъективные мнения.

Библиография

Научные статьи или веб-страницы научных или образовательных ресурсов могут включать библиографию печатных работ. Убедитесь, что это не самоцитирование (автора или источника), включены библиографические сведения о научных книгах, а не только популярные источники, актуальны ли использованные источники? Высокое качество библиографических записей указывает на высокий качественный сайт.

Дата

Сайт может хорошо выглядеть, но если дата его последнего обновления устарела на год, то и информация может быть устаревшей. В зависимости от темы исследования, результаты последних исследований и нынешние события могут оказаться решающими. Это особенно верно, когда речь идет о научно-технических и экономических вопросах.

Гиперссылки

Гиперссылки должны дополнять страницу, добавив контекстную поддержку, более детальную информацию, альтернативную точки зрения, смежные темы.

Проверьте надежность гиперссылок:

Если они укажут вам на другие научные ресурсы и профессиональных организаций, то страницы, которые вы используете, авторитетны, если они ведут к коммерческим сайтам, то надо быть осторожными. Большое количество "мертвых" ссылок указывают на не надежный ресурс.

Внутренние доказательства

Определена ли целевая аудитория и задачи ресурса. Они соответствуют вашим ожиданиям? Изучите название и краткое содержание (если таковое предусматривается), основные идеи, содержание, стиль изложения и научность, профессиональная точность, грамотность.

Доступность

Хороший ресурс доступен во всех своих частях. Барьеры к доступу могут создавать следующие элементы:

- специализированное программное обеспечение и модули подключения,
- использование графических или звуковых файлов,
- формат нецензурных и жаргонных выражений
- вознаграждения или специальной регистрации.

Будьте внимательны по отношению к любому объекту, который требует паспортных данных или любой другой конфиденциальной информации о пользователе кредитной карты.

Оформление

Общее оформления сайта должен отражать качественный веб дизайн: графический дизайн, навигация, доступность. Возможность избежать электронной почты и дискуссионных групп.

2. Выполнить задание.

2.1. Протестировать предложенные веб-страницы:
<http://www.ncfu.ru>
<https://www.aviasales.ru/>
<http://budgetrf.ru/>
<http://www.management.com.ua/>
<http://www.sci.aha.ru/>
<http://www.nstu.ru>
<http://gov.ru>

<http://www.ru.emb-japan.go.jp/>
<http://www.spsl.nsc.ru/>
<http://www.tomsk.ru/>
<http://www.ras.ru/>
<http://www.hse.ru/>
<http://www.stanford.edu/>
<http://www.cfin.ru/>
<http://www.berkeley.edu/>
<http://www.philosophy.ru/>
<http://www.ncfu.ru>

Таблица 1. Качество веб-ресурса

1. Достоверность Web ресурса		2. Точность	3. Управление	4. Авторитетность	5. Объективность	6. Оперативность	7. Актуальность	8. Удобство	9. Доступность	10. Сочетание всех
Автор (ответственное лицо) и возможность связи	Автор (ответственный) адекватен данному ресурсу?	Имеется возможность связи с автором (ответственным лицом): электронная почта или контактный адрес / телефон.	Автор (ответственный) и веб-мастер разделены	Чем подтверждена авторитетность авторов (ответственных) ресурса	Имеются ли ссылки на альтернативные мнения (источники)	Не является ли данный ресурс скрытой рекламой	Является ли информация на странице устаревшей?	Имеется карта сайта	Необходимость дополнительных регистраций	Рейтинг по 10 балльной системе
Заявлена цель ресурса. Она соответствует вашим ожиданиям	Заявлена цель ресурса. Она соответствует вашим ожиданиям		Домен. Регистрация домена ресурса адекватна его целям и задачам							

Порядок выполнения:

1. Запустить браузер Интернет (любой)
2. Зайти на страницы веб-ресурсов, предложенные в задании
3. Оценить качество каждого ресурса по показателям:

Достоверность Web ресурса
 Точность
 Управление
 Авторитетность
 Объективность
 Оперативность

Актуальность
Удобство
Доступность
Сочетание всех параметров

4. Заполнить таблицу "Качество веб-ресурса": проставить рейтинг сайтов.
5. Создать отчет о выполнении лабораторной работы и отправить письмо с отчетом на почту zamira.ncfu@gmail.com.

ПРАКТИЧЕСКАЯ РАБОТА №2. ТЕХНОЛОГИЧЕСКИЕ ПРОЦЕССЫ ОБРАБОТКИ ИНФОРМАЦИИ В ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЯХ. СОЗДАНИЕ ЛИЧНОЙ СТРАНИЦЫ СТУДЕНТА С ПОМОЩЬЮ СЕРВИСОВ GOOGLE

Цель: изучить основные возможности сервиса Google «Сайты».

Разработать личную страницу,
используя сервис Google «Сайты».

Теоретический материал

Сервис Сайты. Этот сервис предназначен для создания мини-сайтов для совместной работы и централизованного хранения связанных между собой документов. В качестве примеров использования сервиса Сайты могут служить, например, сайт отдельного факультета/подразделения учебного заведения или wiki приоритетного научного проекта. Для создания собственных сайтов образовательному учреждению выделяется 10 ГБ дискового пространства в облаке, плюс 500 МБ для хранения файлов каждого сотрудника.

Чтобы создать новый сайт, необходимо перейти в сервис Сайты и выбрать Create. Проще всего создать сайт на основе готового шаблона, который можно выбрать из внушительного списка, представленного в галерее шаблонов.

С помощью стрелок подробно показан процесс создания нового сайта из шаблона. Порядок выполнения действия показан нумерацией соответствующих стрелок. После выбора шаблона и ввода названия сайта

необходимо нажать «Create»; для запуска процедуры генерации сайта. Генерация сайта на основе шаблона может занять одну-две минуты.

Как только сайт будет создан, появится возможность редактировать существующие страницы и добавлять новые.

Процесс редактирования веб-страниц не вызывает особых сложностей, поскольку используется тот же самый интерфейс, что и при редактировании документов в сервисе «Documents».

Предоставление общего доступа к сайту осуществляется так же, как и предоставление общего доступа к документам – необходимо выбрать пункт «Share» интерфейса, затем настроить видимость и права доступа к сайту.

Выполнить задание

1. Зайти под своим профилем в сервисы Google. Выбрать сервис «Сайты»
2. Создать 5 страниц с простым шаблоном: Обо мне, Моя группа, Мои увлечения,
Мои друзья, Моя учеба.
3. Посмотреть возможности изменения формата страниц
4. Посмотреть возможности изменения шаблона страниц

Отчетность по практической работе

Необходимо написать отчет и подвесить в своем профиле в систему университета Ecampus.ncfu.ru (и/или отправить на адрес zamira.ncfu@gmail.com) детальное сообщение о проделанной работе, прикрепив ссылку на созданный сайт с помощью средств google.

В строке 'Тема' письма, указать номер лабораторной работы и ваше ФИО.

Письмо должно содержать приветствие, что выполнено - по пунктам и что не выполнено, подпись автора.

ПРАКТИЧЕСКАЯ РАБОТА №3. МУЛЬТИМЕДИА (4 Ч.)

Цель: изучить технологию размещения мультимедийных файлов на сайте

Теоретическое обоснование

Графика на Web-страницах появилась достаточно давно. Предназначенный для этого тег появился еще в версии 3.2 языка HTML, которая вышла в 1997 году. С тех пор Всемирную паутину захлестнула волна интернет-графики (к настоящему времени, надо сказать, поутихшая).

Как уже говорилось, графические изображения — суть внедренные элементы Web-страниц. Это значит, что они сохраняются в отдельных от самой Web-страницы файлах.

Форматы интернет-графики

На данный момент существует несколько десятков форматов хранения графики в файлах. Но Web-обозреватели поддерживают далеко не все. В WWW сейчас используются всего три формата, которые мы далее рассмотрим.

Нужно отметить, что все три формата поддерживают сжатие графической информации. Благодаря сжатию размер графического файла сильно уменьшается, и поэтому он передается по сети быстрее, чем несжатый файл.

Формат GIF (Graphics Interchange Format, формат обмена графикой) — старожил среди "сетевых" форматов графики. Он был разработан еще в 1987 году и модернизирован в 1989 году. На данный момент он считается устаревшим, но все еще широко распространен.

Достоинств у него довольно много. Во-первых, GIF позволяет задать для изображения "прозрачный" цвет; закрашенные этим цветом области изображения станут своего рода "дырками", сквозь которые будет "просвечивать" фон родительского элемента. Во-вторых, в одном GIF-файле можно хранить сразу несколько изображений, фактически — настоящий

фильм (анимированный GIF). В-третьих, из-за особенностей применяемого в нем сжатия он отлично подходит для хранения штриховых изображений (карт, схем, рисунков карандашом и пр.).

Недостаток у формата GIF всего один — он совершенно не годится для хранения полутоновых изображений (фотографий, картин и т. п.). Это обусловлено тем, что GIF-изображения могут включать всего 256 цветов, и потерями качества при сжатии.

GIF используется для хранения элементов оформления Web-страниц (всяческих линеек, маркеров списков и т. п.) и штриховых изображений.

Формат JPEG (Joint Photographic Experts Group, Объединенная группа экспертов по фотографии) был разработан в 1993 году специально для хранения полутоновых изображений. Для чего активно применяется до сих пор.

JPEG, в отличие от GIF, не ограничивает количество цветов у изображения, а реализованное в нем сжатие лучше всего подходит для полутоновых изображений. Однако он плохо справляется с штриховой графикой, не поддерживает "прозрачный" цвет и анимацию.

Формат PNG (Portable Network Graphics, перемещаемая сетевая графика) появился на свет в 1996 году. Он разрабатывался как замена устаревшему и не очень удобному GIF, а также, в некоторой степени, JPEG. В настоящее время он последовательно отвоевывает "жизненное пространство" у GIF.

К достоинствам формата PNG можно отнести возможность хранения как штриховых, так и полутоновых изображений и поддержку полупрозрачности. Недостаток всего один и не критичный — невозможность хранения анимации.

Осталось назвать расширения, под которыми сохраняются файлы того или иного формата. Файлы GIF и PNG имеют "говорящие" расширения gif и png, а файлы JPEG — jpg, jpe или jpeg.

Вставка графических изображений

Добавить на Web-страницу графическое изображение позволяет одинарный тег . Web-обозреватель поместит изображение в том месте Web-страницы, в котором встретился тег .

Обязательный атрибут тега SRC служит для указания интернет-адреса файла с изображением.

Пример:

```
<IMG SRC="image.gif">
```

Этот тег помещает на Web-страницу изображение, хранящееся в файле image.gif, который находится в той же папке, что и файл самой этой Web-страницы.

Пример:

```
<IMG SRC="/images/picture.jpg">
```

Данный тег помещает на Web-страницу изображение, хранящееся в файле picture.jpg, который находится в папке images, вложенной в корневую папку Web-сайта.

Пример:

```
<IMG SRC="http://www.othersite.ru/book12/author.jpg">
```

А этот тег помещает на Web-страницу изображение, хранящееся в файле с интернет-адресом <http://www.othersite.ru/book12/author.jpg>, т. е. изображение с другого Web-сайта.

Рассмотрим еще один атрибут тега , который может нам пригодиться в дальнейшем.

Поскольку изображение хранится в отдельном от Web-страницы файле, Web-обозревателю придется послать Web-серверу еще один запрос на его получение. Web-серверу нужно найти этот файл и отправить его Web-обозревателю. Файл должен загрузиться по сети.

На все это требуется время. Если изображений на Web-странице много, все они велики по размеру, а канал связи медленный, понадобится

значительное время. Может случиться так, что сама Web-страница будет успешно загружена и отображена на экране, а изображения - еще нет. И Web-обозреватель вместо не загруженного еще изображения выведет на экран пустой прямоугольник.

Возникают две проблемы. Во-первых, пустые прямоугольники вместо изображений выглядят некрасиво. Во-вторых, посетитель не сможет понять, что за изображение должно находиться вместо того или иного прямоугольника, и стоит ли ждать окончания его загрузки.

И если с первой проблемой справиться практически невозможно, то вторую мы вполне способны решить. Для этого тег `<IMG>` поддерживает необязательный атрибут `ALT`, с помощью которого указывается так называемый текст замены. Он будет отображаться в пустом прямоугольнике, обозначающем незагруженное изображение, пока это изображение не загрузится:

```
<P><IMG SRC="image.gif" ALT="Пример изображения"></P>
```

Вот еще несколько необязательных атрибутов:

- `height`-Высота изображения
- `width`-Ширина изображения
- `align`-Определяет как рисунок будет выравниваться по краю и способ обтекания текстом

Мультимедиа. Форматы файлов и форматы кодирования

Мультимедиа — это, в первую очередь, аудио и видео. Мультимедиа в приложении к Web-дизайну — это аудио- и видеоролики, размещенные на Web-страницах.

До недавних пор разместить на Web-странице аудио- или видеоролик можно было только с помощью громадного HTML-кода, дополнительных программ и "шаманских плясок" вокруг всего этого. Но сейчас, с появлением HTML 5 и поддерживающих его (хотя бы частично) Web-обозревателей,

потребуется всего один тег. Какой? Очень простой, не сложнее уже знакомого нам тега !

Форматы файлов и форматы кодирования

Форматов мультимедийных файлов существует не меньше, чем форматов файлов графических. Как и в случае с интернет-графикой, Web-обозреватели поддерживают далеко не все мультимедийные форматы, а только немногие. (Хотелось бы автору посмотреть на Web-обозреватель, который поддерживает все форматы файлов — и на сам Web-обозреватель, и на его размеры...)

Но Web-обозревателю мало поддерживать только сам формат мультимедийных файлов. Он должен быть "знаком" и с форматом кодирования записанной в нем аудио- и (или) видеоинформации. В мире мультимедиа так — разные файлы одного формата могут хранить информацию, закодированную разными форматами. Более того, аудио- и видеодорожки мультимедийного файла практически всегда кодируются разными форматами.

Практически все форматы кодирования мультимедийных данных поддерживают их сжатие. Благодаря этому размер мультимедийных файлов значительно (иногда на несколько порядков) уменьшается, что благотворно сказывается на скорости их передачи по сети.

Перечислим и кратко опишем все форматы мультимедийных файлов, используемые в Web-дизайне и поддерживаемые Web-обозревателями.

- WAV (WAVE, волна) — "старожил" среди мультимедийных форматов. Был разработан Microsoft в самом начале 90-х годов прошлого века для хранения аудио-данных и применяется для этой цели до сих пор. Файлы такого формата имеют расширение wav.

- OGG — более новый формат. Был разработан около десяти лет назад некоммерческой организацией Xiph.org для хранения аудио- и видеоинформации. Файлы этого формата имеют расширения ogg

(универсальное расширение), oga (аудио-файлы) и ogv (видеофайлы); последние два расширения встречаются редко.

- MP4 — также "новичок". Был разработан организацией Motion Picture Expert Group (Экспертная группа по вопросам движущегося изображения; также известна как MPEG) в 1998 году для хранения аудио- и видеоданных. Файлы этого формата имеют расширение mp4.

- QuickTime — формат очень старый, он старше даже WAV. Был разработан Apple в 1989 году для хранения аудио- и видеоданных. Файлы такого формата имеют расширение mov.

Теперь рассмотрим форматы кодирования аудио и видео, поддерживаемые современными Web-обозревателями.

- PCM (Pulse-Coded Modulation, импульсно-кодовая модуляция) — самый простой и самый старый формат кодирования. Он даже не поддерживает сжатие информации. Служит для кодирования аудиоданных.

- Vorbis — более современный формат кодирования. Был представлен организацией Xiph.org (разработчиком формата файла OGG) в 2002 году. Используется для кодирования аудиоданных.

- AAC (Advanced Audio Coding, развитое кодирование аудио) — не очень новый формат кодирования. Был разработан организацией Motion Picture Expert Group в 1997 году. Применяется для кодирования аудиоданных.

- Theora — пожалуй, самый "молодой" формат кодирования. Он также был разработан организацией Xiph.org несколько лет назад. Используется для кодирования видеоданных.

- H.264 — тоже очень "молод". Был представлен организациями Motion Picture Expert Group и Video Coding Experts Group (Группа экспертов по кодированию видео) в 2003 году. Предназначен для кодирования видеоданных.

Почти все эти форматы являются открытыми. Исключения — формат файлов QuickTime, принадлежащий Apple, и формат кодирования H.264, защищенный более чем сотней патентов.

Типы MIME

По сети передаются самые разные данные: Web-страницы, графические изображения, аудио- и видеофайлы, архивы, исполняемые файлы и пр. Эти данные предназначены разным программам. К тому же, с разными данными программа, принявшая их, может поступить по-разному. Так, Web-обозреватель при получении Web-страницы или графического изображения отобразит их на экране, а при получении архива или исполняемого файла — откроет или сохранит его на диске.

Всем передаваемым по сети данным присваивается особое обозначение, однозначно указывающее на их природу, — тип MIME (Multipurpose Internet Mail Extensions, многоцелевые расширения почты Интернета). Тип MIME присваивает данным программа, их отправляющая, например, Web-сервер. А принимающая программа (тот же Web-обозреватель) по типу MIME принятых данных определяет, поддерживает ли она эти данные, и, если поддерживает, что с ними делать.

Вставка аудиоролика

Для вставки на Web-страницу аудиоролика язык HTML 5 предусматривает парный тег `<AUDIO>`. Интернет-адрес файла, в котором хранится этот аудиоролик, указывают с помощью атрибута SRC этого тега: `<AUDIO SRC="sound.wav"></AUDIO>`

Встретив тег `<AUDIO>`, Web-обозреватель может сразу загрузить и воспроизвести аудиофайл, только загрузить его без воспроизведения или вообще ничего не делать. Также он может вывести на Web-страницу элементы управления, с помощью которых посетитель запускает воспроизведение аудиофайла, приостанавливает его, прокручивает вперед или назад и регулирует громкость. Все это настраивается с помощью различных атрибутов тега `<AUDIO>`.

Тег `<AUDIO>` создает блочный элемент Web-страницы. Так что мы не сможем вставить аудиоролик на Web-страницу в качестве части абзаца. Зато,

чтобы поместить его в отдельный абзац, нам не придется совершать никаких дополнительных действий (в отличие от изображения).

По умолчанию Web-обозреватель не будет воспроизводить аудиоролик. Чтобы он это сделал, в теге `<AUDIO>` нужно указать особый атрибут `AUTOPLAY`. Это действительно особый атрибут: он не имеет значения — достаточно одного его присутствия в теге, чтобы он начал действовать (атрибут тега без значения):

```
<P>Сейчас вы услышите звук!</P>  
<AUDIO SRC="sound.ogg" AUTOPLAY></AUDIO>
```

По умолчанию аудиоролик никак не отображается на Web-странице (что, впрочем, понятно — аудио нужно не смотреть, а слушать). Но если в теге `<AUDIO>` поставить атрибут без значения `CONTROLS`, Web-обозреватель выведет в том месте Web-страницы, где проставлен тег `<AUDIO>`, элементы управления воспроизведением аудиоролика. Они включают кнопку запуска и приостановки воспроизведения, шкалу воспроизведения и регулятор громкости:

```
<P>Нажмите кнопку воспроизведения, чтобы услышать звук.</P>  
<AUDIO SRC="sound.ogg" CONTROLS></AUDIO>
```

Атрибут без значения `AUTOBUFFER` имеет смысл указывать в теге `<AUDIO>` только в том случае, если там отсутствует атрибут `AUTOPLAY`. Если он указан, Web-обозреватель сразу после загрузки Web-страницы начнет загружать файл аудиоролика — это позволит исключить задержку файла перед началом его воспроизведения.

Вставка видеоролика

Для вставки на Web-страницу видеоролика предназначен парный тег `<VIDEO>`. Интернет-адрес видеофайла указывается с помощью знакомого нам атрибута `SRC` этого тега:

```
<VIDEO SRC="film.ogg"></VIDEO>
```

Встретив этот тег, Web-обозреватель выведет в том месте Web-страницы, где он проставлен, панель просмотра содержимого видеоролика. В зависимости от указанных нами в теге <VIDEO> атрибутов, он может сразу загрузить и воспроизвести аудиофайл, только загрузить его без воспроизведения или вообще ничего не делать. Также он может вывести на Web-страницу элементы управления воспроизведением видеоролика.

Как и тег <AUDIO>, тег <VIDEO> создает блочный элемент Web-страницы и поддерживает атрибуты AUTOPLAY, CONTROLS и AUTOBUFFER:

```
<VIDEO SRC="film.ogg" AUTOPLAY CONTROLS></VIDEO>
```

Если воспроизведение видеоролика еще не запущено, в панели просмотра будет выведен первый его кадр или вообще ничего (конкретное поведение зависит от Web-обозревателя). Но мы можем указать графическое изображение, которое будет туда выведено в качестве заставки. Для этого служит атрибут POSTER тега <VIDEO> ; его значение указывает интернет-адрес нужного графического файла:

```
<VIDEO SRC="film.ogg" CONTROLS POSTER="filmposter.jpg"></VIDEO>
```

Здесь мы указали для видеоролика изображение-заставку, которое будет выведено в панели просмотра перед началом его воспроизведения и которое хранится в файле filmposter.jpg.

Выполните задания

1. Добавить в web-страницу 2 аудиоролика (файлы прикреплены).

Используя знания, полученные из лекции, создайте web-страницу, которая содержала бы аудио материалы.

Чтобы сохранить аудиозапись на своем компьютере нужно: Нажать на ссылку правой кнопкой мыши, из Контекстного меню выбрать "Сохранить объект как...", выбрать папку своей группы и сохранить туда медиафайл.

2. Добавить в web-страницу 2 видеоролика (файлы прикреплены).

Используя знания, полученные из лекции, создайте web-страницу, которая содержала бы видео материалы.

Чтобы сохранить ролик на своем компьютере нужно: Нажать на ссылку правой кнопкой мыши, из Контекстного меню выбрать "Сохранить объект как...", выбрать папку своей группы и сохранить туда медиафайл.

3. Добавить в web-страницу 3 фото студентов своей группы (файлы можно загрузить самостоятельно)

Отчетность по практической работе

Необходимо написать отчет и подвесить в своем профиле в систему университета Ecampus.ncfu.ru (и/или zamira.ncfu@gmail.com отправить на адрес zamira.ncfu@gmail.com) детальное сообщение о проделанной работе, прикрепив архив с созданными страницами (страницей) с размещенными аудио, видео и фотофайлами.

В строке 'Тема' письма, указать номер практической работы и ваше ФИО.

Письмо должно содержать приветствие, что выполнено - по пунктам и что не выполнено, подпись автора.

ПРАКТИЧЕСКАЯ РАБОТА №4. ОСНОВЫ JAVASCRIPT

Цель: познакомиться с основами JavaScript и сформировать первичные навыки написания скриптов для сайта

Теоретическое обоснование

Переменные

Имя (идентификатор) переменной может состоять из букв, цифр и знаков \$ и _ (подчеркивание). Имя переменной не должно начинаться с цифры. В имени различаются строчные (маленькие) и прописные (большие) буквы. Например, *Alfa*, *alfa* и *aLFa* – разные переменные. В качестве имён переменных нельзя использовать ключевые слова языка JavaScript. Переменные могут иметь следующие типы:

- целые числа (integer);

- числа с плавающей запятой (float);
- строки (string);
- логические (booleans);
- объекты.

В JavaScript тип переменной явно не объявляется. Создают переменную чаще всего, присваивая ей начальное значение:

```
x = 2; // или var a = 2
```

В приведённом примере две косые черты (//) указывают на комментарий. Начало и конец многострочного комментария обозначаются символами /* и */. После оператора можно ставить точку с запятой. Если в одной строке записано несколько операторов, то точка с запятой отделяет операторы друг от друга. Можно создать переменную без присвоения ей начального значения, например,

```
Var y;
```

Ключевое слово *var* используется для определения области видимости переменных. Переменная, созданная в функции с использованием *var*, видна только в этой функции и называется локальной. Во всех остальных случаях переменная является глобальной и видна во всей программе, включая функции. Пример:

```
<HTML>Скрипт
<SCRIPT>      var      x      =      2      //глобальная      переменная
y      =      3*x      //глобальная      переменная
z=kvadr(y)
alert("z="+z+" x="+x)//z=36x=2
functionkvadr(a)
{ var x = a*a //x - локальная переменная, действующая
//только в kvadr(a)
returnx//x=36
}
```

</script>

</html>

Тип переменной определяется (почти так же как в языках Perl и PHP) из контекста программы. В приведённом примере использованы операторы присваивания в стиле языка C (подробнее в разделе Операторы).

```
1.      a      =      2;      //      целое
2.  a  +  =  1.5  //  2  +  1.5  =  3.5  -  число  с  плавающей  запятой
3.      a      *      = "4"      // 3.5*4=14
4.  a  +  = "2" //конкатенация строковых величин "14" и "2".
      //Результат "142"
5.      b      =      53;      //целое      число
6.      c      =      "      года.      ";
7.  d = b + c //строка "53 года. "
```

В строках 4 и 7 знак + использован для обозначения операции конкатенации (слияния) строковых переменных, а не сложения двух чисел. Одинаковое обозначение двух разных операций приводит к необходимости использовать функции явного преобразования переменной из строкового типа в тип числа с плавающей запятой (функция *parseFloat()*) или в тип целого числа (функция *parseInt()*). Пример использования функций *parseFloat()* и *parseInt()*:

```
1.      a      = "3.7";
2.      b      =      "6.1";
3.  c  =  a  *  b  //  результат  3.7  *  6.1  =  22.57
4.      c=a+b;      //      результат      "3.76.1"
5.  d=parseFloat(a)+parseFloat(b); //  результат  3.7  +  6.1=9.8
6.  e=parseInt(a)+parseInt(b); // результат 3 + 6 = 9
```

В строке 3 переменные *a* и *b* перемножаются и поэтому автоматически преобразуются в числа с плавающей запятой. В строке 4 выполняется операции конкатенации, так как в строках 1 и 2 переменным *a* и *b* были

присвоены строковые значения. В строках 5 и 6 явно задаётся преобразование переменных *a* и *b* в числовой тип, поэтому выполняется операция сложения.

Операторы

В JavaScript, как и в большинстве языков программирования, используются традиционные математические операторы и операторы, применяемые только в программировании. Математические операторы делятся на три группы: арифметические, операторы сравнения и логические. Среди арифметических операторов (табл. 1) в JavaScript отсутствует возведение в степень.

Таблица 1. Арифметические операторы		
Название	Обозначение	Пример
Сложение	+	$a + b$
Вычитание	-	$a - b$
Умножение	*	$a * b$
Деление	./	a / b
Деление по модулю	%	$a \% b$
Увеличение на 1 (инкремент)	++	$a++$
Уменьшение на 1 (декремент)	--	$a--$

Возведение числа *A* в степень *N* выполняется так:

x $=$ `Math.pow(A,N)`,

где `pow(A,N)` – метод объекта `Math`.

Операторы сравнения (табл. 2) применяются к числовым, строковым и логическим выражениям.

Таблица 2. Операторы сравнения		
Название	Обозначение	Пример

Равно	==	a == b
Не равно	!=	a != b
Больше	>	a > b
Меньше	<	a < b
Больше или равно	>=	a >= b
Меньше или равно	<=	a <= b

В JavaScript используются три логических оператора: И, ИЛИ и НЕ, имеющие обозначения &&, || и ! соответственно. Результат логической операции может иметь одно из двух значений: true (истина) и false (ложь). Логической величине true соответствует число 1, а false – 0. Примеры логических операций:

```

a = 2; b = 3; st = "a"; // исх. данные для лог.выражений
a > b || st == "a" // истина
a > b || st > "a" // ложь
a > b && st == "a" // ложь
a < b && st == "a" // истина
!(a < b) // ложь

```

Операторы, применяемые только в программировании, также можно разделить на три группы: присваивания, условного перехода и циклические.

Примеры операторов присваивания:

```

a = 5 //записать число 5 в область памяти,
//отведённую под переменную a
a + = 2 //вместо оператора a = a + 5
a - = 2;
a * = 2;

```

```

a / = 2;
a %= 2

```

Результаты действия инкремента и декремента зависят от того, как они расположены относительно операнда:

```

a = b++; // сначала a присваивается значение b, затем b увеличивается на 1

```

```

a = ++b; // сначала b увеличивается на 1, затем a присваивается значение b

```

Примеры операторов условного перехода if и switch:

```

Var a=3
If (a > 2) alert("a больше двух");
a = 1;
b = -3;
c = 2;
det = Math.pow(b,2) - 4*a*c
if (det < 0)
{
    alert("действительных корней нет")
    r = 0
}
else if (det == 0)
{
    alert("корни одинаковые")
    r=1
}
else
{
    alert("два корня")
    r=2
}

switch (a)
{
    case 0:
        alert("ноль")
        break

```

```

case 1:
    alert(1)
    break
case 2:
    alert("два")
    break
default:
    alert("не подходит");
}

```

Если в операторе *switch* убрать ключевое слово *break*, то будут выполнены все операторы после первого совпадения.

Вместо условного оператора *if* может использоваться следующая конструкция:

переменная = (условие)? значение 1 : значение 2

Например, в результате выполнения оператора

`x = (4 > 2) ? "правильно" : "ошибка"`

переменная *x* примет значение правильно.

Примеры применения операторов цикла *for*, *while* и *do while*

```

// Оператор for. Сложение всех нечётных чисел от 1 до 10
S = 0;
for(i=1; i<=10; i++) {S+=i}
S //сложение чисел 5,10,...,25
for(i=5; i<=25; i+=5) {S+=i}
// Оператор while. Сложение всех нечётных чисел от 1 до 10
S = 0; i = 0;
while(i < 10) {S+=++i}
// Оператор do ... while.
//Сложение всех нечётных чисел от 1 до 10
S = 0; i = 0;
do

```

```
{
                                S
                                +=
                                ++i}
while(i < 10)
```

Массивы

Массивы в JavaScript одновременно рассматриваются и как объекты и как массивы в традиционном понимании. Способы создания массивов показаны в следующих примерах:

```
var a = new Array(5) //создан массив из пяти элементов
var b = new Array(2,7,12) // создан массив b из трёх
//элементов и его элементам присвоены значения
```

Нумерация элементов массива начинается с нуля. Для обращения к элементу массива используются квадратные скобки, например

```
x = b[1] // x равен 7
```

Длина массива определяется как его свойство:

```
c = a.length // длина массива a c= 5
```

Элементы одного массива могут иметь разный тип. В следующем примере массив состоит из строковых и числовых элементов.

```
earth = new Array(4)
earth[0]= "Планета" // строка
earth[1]= "24 часа" // строка
earth[2]= 6378 // целое число
earth[3]= 365.25 // число с плавающей запятой
```

Информацию о планете *Земля* удобно хранить в ассоциативном массиве.

```
earthA = new Array(4)
earthA.type_ = "Планета"
earthA.day_ = "24 часа"
earthA.radius= 6378
earthA.period = 365.25
```

Слова *type* и *day* – ключевые в JavaScript. Поэтому к идентификаторам переменных *type_* и *day_* добавлен знак подчёркивания. Для извлечения из обоих массивов радиуса Земли служат операторы:

```
R=earth[2]
```

```
R= earthA.radius
```

Элементами массива могут служить массивы:

```
product = new Array()
product[0] = new Array("Пшеница",250)
product[1] = new Array("Рожь",300)
product[2] = new Array("Ячмень",200)
```

Массив `product` – двумерный. Элемент `product[0][1]` имеет значение 250.

Длина двумерного массива определяется свойством `length`:

```
L = product.length // L = 3
m = product[0].length // m = 2
```

Элементами массива `product` являются три массива, каждый из которых состоит из двух элементов.

Строки

Строка в JavaScript является одновременно и объектом `string` и переменной, поэтому может быть создана двумя способами:

```
st1 = new String("Строка – это объект")
st2 = "Строка – это переменная"
```

Строки, рассматривая их как переменные, можно объединять с помощью операции конкатенации. Рассмотрим пример, в котором новая строка получается в результате объединения двух строковых переменных, строковой константы и числовой переменной:

```
st1 = "Санкт-Петербург основан"
st2 = "в 1703 году"
st3 = st1 + " в " + n + st2
```

В строке `st3` будет предложение Санкт-Петербург основан в 1703 году. Тип переменной `n` изменён автоматически. Наиболее часто над строкой совершаются следующие действия:

- ищется подстрока;

- выделяется несколько символов (подстрока), начиная с символа с заданным номером;
- заменяется одна подстрока другой.

Для поиска подстроки используется метод *indexOf()*, имеющий следующий синтаксис:

строка.indexOf(подстрока [начало]) ,

//где начало – номер символа в строке, с которого нужно начинать поиск.

Результат применения метода *indexOf()* – номер символа, с которого начинается искомая подстрока. Нумерация символов начинается с нуля. Найдём в строке *st3* номер символа, с которого начинается подстрока *Петербург*:

n = st3.indexOf("Петербург") // результат – 6

Для выделения нескольких символов используется метод *substr()*, имеющий следующий синтаксис:

строка.substr(начало [длина]) ,

где: начало – номер символа, с которого начинается подстрока, длина – длина искомой подстроки

Результат применения метода – найденная подстрока. Выделим в *st3* слово *Петербург*:

gorod = st3.substr(6,9)

Для замены подстроки используется метод *replace()*, имеющий следующий синтаксис:

строка.replace(регулярное выражение, заменяющая подстрока)

В качестве регулярного выражения в простейшем случае можно применить заменяемую подстроку. Заменяем в строке *st3* подстроку 1703 году на подстроку XVIII веке:

osnovanie=st3.replace(/1703 году/, " XVIII веке")

Строка *st3* примет значение Санкт-Петербург основан в XVIII веке.

Объекты Math и Number

Свойствами объекта *Math* служат математические константы, а методами - математические функции. Примеры использования объекта *Math*:

```

r = 5
//Радиус как объект: r = new Number(5)
L = 2 * Math.PI * r //Длины окружности L=31.41592653589793
Lc = Math.ceil(L) // Lc=32. Округление до ближайшего
// большего или равного целого
Ld = Math.floor(L) // Ld=31.Округление до ближайшего
// меньшего или равного целого
Lok = Math.round(L)//Lok=31. Округление до ближайшего целого
S= Math.PI * Math.pow(r,2) //Площадь круга
```

Все переменные в примере являются объектами Number. Общий вид выражения для создания объекта Number:

переменная = new Number(число)

Создание конкретного объекта:

z = new Number(43.567)

Часто используемые свойства объекта Number:

```

L = new Number(31.41592653589793)
//L.toFixed(n) - n знаков после запятой
L_3 = L.toFixed(3) // L_3=31.416
//L.toPrecision(n) - n значащих цифр
L4 = L.toPrecision(4) //L4=31.42
```

ПРАКТИЧЕСКАЯ РАБОТА №5. ДОСТУП К ОБЪЕКТАМ И СЦЕНАРИЯМ

Цель: научиться использовать различные способы доступа к свойствам и методам объектов для внесения изменений в HTML-документ.

Теоретическое обоснование

Все изменения в HTML-документе производятся через свойства, методы и события объектов, входящих в состав объектной модели документа. Рассмотрим сценарий (пример 1), в котором используются типичные способы доступа к свойствам и методам объектов.

Пример 1.

```
<HTML ID='DOCUM'>
<CENTER><H1 ID='zag'> ВАСЬКА</h1>
Вариант, совместимый с Mozilla<P>
<FORM NAME='fl'>
<INPUT TYPE=BUTTON name="bot" onclick=bm() value='Увеличить'>
</FORM >
<P><IMG SRC="VaskaM.jpg" ID='Vas' onclick=bm()>
</center>
<!--В скрипте закомментирован вывод окна со списком объектов -->
<SCRIPT>
/*
var msg="
for(i=0; i<document.all.length;i++)
msg+=i + ' ' + document.all[i].tagName + ' ID=' + document.all[i].id+'\n'
alert(msg)
*/
flag=true
function bm()
{ if(flag)
```

```

        {
            document.images[0].src='VaskaB.jpg'
            flag=false;
            document.forms[0].bot.value="Уменьшить"
            document.forms[0].bot.style.background='red'
            document.all.bot.style.color='black';
        }
    else
    {
        document.getElementById("Vas").src='VaskaM.jpg';
        //обращение к кнопке по индексу
        document.forms[0].bot.value="Увеличить"
        document.forms[0].bot.style.background='green'
        document.all.bot.style.color='white';
        flag= true
    }
}
</script>
</HTML>

```

Задание 1. Выполнить пример 1

В примере 1 сценарий используется для замены фотографии и изменения цвета кнопки и надписи на ней. Друг друга меняют маленькая и большая фотографии кота Васьки.

Замена фотографии и изменение вида кнопки в примере происходят при щелчке мышкой по кнопке или по самой фотографии. В тегах <INPUT> и помещён атрибут onclick=bm(), при помощи которого в ответ на щелчок мышкой вызывается функция bm(). Такой атрибут называют обработчиком событий. Имя файла с фотографией является свойством SRC объекта IMG. Замена фотографии делается в сценарии двумя равнозначными способами:

```

document.images[0].src='VaskaB.jpg'
document.getElementById("Vas").src='VaskaM.jpg'

```

В первом варианте доступ к фотографии происходит через коллекцию *images*, в которой все расположенные на странице фотографии (в

примере одна фотография) пронумерованы. Нумерация начинается с нуля. Во втором варианте используется метод `getElementById()` объекта `document`. Параметром в методе `getElementById()` служит *ID* (идентификатор) объекта *IMG*. Весь оператор читается так: "свойству *src* объекта, имеющего *ID=Vas* и входящего в состав объекта *document*, присвоить значение *VaskaM.jpg*". Сам объект *document* является дочерним по отношению к объекту *window*. Поэтому полное обращение к свойству *src* следовало бы писать так:

```
window.document.images[0].src='VaskaB.jpg'  
window.document.getElementById("Vas").src='VaskaM.jpg'
```

Обычно, за исключением тех случаев, когда нужно воспользоваться свойствами или методами самого объекта *window*, это слово опускают.

Более сложная иерархия объектов наблюдается при обращении к свойствам кнопки:

```
document.forms[0].bot.value="Уменьшить"  
document.forms.fl.bot.style.background='red'
```

Первый оператор служит для изменения надписи на кнопке. Доступ к кнопке происходит по её имени (*name='bot'*). Доступ к форме осуществляется через коллекцию *forms* и номер формы в коллекции.

С помощью второго оператора меняется цвет кнопки. Доступ к форме осуществляется через коллекцию *forms* и имя формы *fl*. Своеобразие доступа к свойствам, задаваемым стилем, состоит в том, что *style* выступает формально как дочерний объект. В рассматриваемом примере родительским по отношению к *style* является объект с именем *bot*, то есть, кнопка.

Для обращения к объектам удобна коллекция *all* объекта *document*. В коллекцию *all* входят все объекты HTML-документа, поэтому выбирать объект нужно по его *ID* или имени. Коллекция *all* использована для изменения цвета надписи на кнопке:

```
document.all.bot.style.color='black'
```

Все описанные способы доступа к объектам HTML-документа базируются на объектной модели документа (DOM). Наиболее трудной задачей, стоящей перед разработчиком сценариев, является разработка сайтов, которые будут одинаково выполняться на всех браузерах. Рассматриваемый пример HTML-документа подобран так, что он (почти) одинаково выполняется на браузерах Internet Explorer 6.0 и Mozilla Firefox 2.0, несмотря на то, что объектные модели в этих браузерах отличаются друг от друга. Для того, чтобы убедиться в этом уберите символы комментариев /* и */ из первой и пятой строчек скрипта и откройте страницу в обоих браузерах. Появится окно со списком всех объектов, созданных браузером для открытого документа. В созданной браузером Internet Explorer модели – 12 объектов, а у Mozilla Firefox – 10.

Рассмотрим подробнее раскомментированный фрагмент скрипта.

```
var                                                    msg=""
for(i=0;                                                    i<document.all.length;i++)
msg+=i + ' ' + document.all[i].tagName + ' ID=' + document.all[i].id+'\n'
alert(msg)
```

Коллекция *all* всех объектов HTML-документа имеет свойство *length*, равное количеству всех объектов модели документа. Любой объект имеет свойство *tagName*, значение которого совпадает с именем объекта. В приведённом фрагменте скрипта в цикле формируется строковая переменная *msg*, состоящая из номеров, имён и *ID* объектов. Сочетание символов *\n* служит для перевода строки.

Метод `getElementById()` - входит в стандарт W3C и поддерживается всеми браузерами. Коллекция *all* разработана фирмой Microsoft для браузера Internet Explorer. В настоящее время эта коллекция поддерживается почти всеми браузерами. Как ни странно, в новом браузере Edge, встроенном в ОС Windows 10, коллекция *all* не поддерживается.

Задание 2.

Напишите HTML-документ, который в окне браузера отображается в виде следующих трёх строк:

- ДОСТУП К СВОЙСТВАМ И МЕТОДАМ
- Коллекция all
- Метод `getElementById()`

Первую строку поместите в контейнер `<H2>...</h2>`, вторую – в контейнер `<P> ...</p>`, третью – в контейнер `<DIV> ... </div>`. Напишите скрипт для изменения цветов фона и букв надписей, при щелчке по этим строкам. При щелчке по первой строке цвет букв должен меняться с чёрного на белый или с белого на чёрный, а фон – с жёлтого на синий или с синего на жёлтый. Так же должны меняться цвета третьей строки.

При щелчках по второй строке цвет букв на ней должен меняться с красного на белый и наоборот, а цвет фона - с белого на зелёный и наоборот.

Для изменения первой и третьей строк примените метод `getElementById()`, а для второй строки – коллекцию `all`.

Сценарий в HTML-документе можно размещать тремя способами:

- в открывающем теге в качестве значения атрибута событие (см. пример 2.1);
- в контейнере, ограниченном тегами `<SCRIPT> ...</script>`;
- в отдельном файле.

Контейнеры со сценариями могут размещаться в любом месте HTML-документа. Количество сценариев в одном HTML-документе не ограничено.

Выполнение сценариев происходит при загрузке HTML-документа и при наступлении событий. Во время загрузки HTML-документа браузер последовательно просматривает встречающиеся сценарии и пытается их выполнить. Если встретился вызов функции, то браузер ищет её описание в текущем и во всех ранее загруженных сценариях. Поэтому описание функции должно размещаться либо в одном сценарии с вызовом, либо в сценариях расположенных выше вызова функции.

Сценарий можно хранить в отдельном файле. Для вставки сценария в HTML-документ служит атрибут *SRC* тега *<SCRIPT>*. Сценарий, размещённый в отдельном файле, можно использовать на многих страницах сайта. В примере 1.2 рассматривается одна из страниц сайта сети магазинов.

Пример 2

```
html><!-- СКРИПТ загружается из файла primJs.js--> <HEAD>
<TITLE>Сеть</title>
</head>
<body>
<SCRIPT          language="JavaScript"          src="PrimJs.js">
</script>
<H2 align=center style="color:green">Магазин "ПОДАРКИ"</h2>
Адрес:          Лесная          ул.,          д.2<P>
Транспорт:      трамваи      7,      23,      автобусы      56,      93
</body>
</html>

//          Файл          primJs.js
a="background-color:#00ffff;          color:#ff00ff;"
a+="font-size:24pt;          font-family:'Times          New          Roman'"
naim='Сеть          магазинов          "ВСЁ          ДЛЯ          ДОМА"'
var          da=new          Date()
d=da.getDate()+"."+(da.getMonth()+1)+"."+da.getFullYear()
document.write('<P          align=center          style=          "'+a+'">'+
naim+'</p><P>Сегодня '+d+'</p>')
```

Задание 3. Выполнить пример 2

Общие для страниц всех магазинов сети заголовки с названием сети и сегодняшняя дата формируются скриптом, размещённым в отдельном файле *primJs.js*. Метод *write(HTML-код)* служит для вставки в страницу размеченного текста (HTML-кода). Этот метод применяется только при

загрузке страницы, так как его применение после окончания загрузки приведёт к стиранию старого содержимого страницы.

Задача 1.2. Добавьте в пример 1.2 три страницы. Две страницы должны отображать информацию о магазинах Посуда и Мебель. Третья страница – главная в сайте сети магазинов ВСЁ ДЛЯ ДОМА. На ней должны быть ссылки на страницы магазинов, входящих в сеть. На страницах магазинов должен использоваться скрипт из файла *primJs.js*.

ПРАКТИЧЕСКАЯ РАБОТА №6. НАСТРОЙКА НАЧАЛЬНЫХ ПАРАМЕТРОВ КОММУТАТОРА

Цель: научиться проводить настройку начальных параметров коммутатора

Теоретическое обоснование

Общие сведения

В этом упражнении вы осуществите базовую настройку коммутатора. Затем вам будет необходимо обеспечить безопасность доступа к интерфейсу командной строки (CLI) и портам консоли с помощью зашифрованных и текстовых паролей. Вы также научитесь настраивать сообщения для пользователей, выполняющих вход в систему коммутатора. Эти баннеры также предупреждают пользователей о том, что несанкционированный доступ запрещен.

Часть 1. Проверка конфигурации коммутатора по умолчанию

Шаг 1. Войдите в привилегированный режим EXEC.

Привилегированный режим EXEC дает доступ ко всем командам коммутатора. Но поскольку многие привилегированные команды задают рабочие параметры, привилегированный доступ должен быть защищен паролем во избежание несанкционированного использования.

Набор команд привилегированного режима EXEC включает команды, которые доступны в пользовательском режиме EXEC, а также команду configure, открывающую доступ к остальным командным режимам.

А. Щелкните S1, а затем вкладку CLI. Нажмите клавишу ввода.

Б. Перейдите в привилегированный режим EXEC, выполнив команду enable.

```
Switch> enable
```

```
Switch#
```

Обратите внимание, что командная строка изменилась, отображая переключение в привилегированный режим EXEC.

Шаг 2. Изучите текущую конфигурацию коммутатора.

А. Введите команду show running-config.

```
Switch# show running-config
```

Б. Ответьте на следующие вопросы.

1. Сколько у коммутатора интерфейсов FastEthernet?
2. Сколько у коммутатора интерфейсов Gigabit Ethernet?
3. Каков диапазон значений, отображаемых в линиях vty?
4. Какая команда отображает текущее содержимое энергонезависимого ОЗУ (NVRAM)?
5. Почему коммутатор отвечает сообщением startup-config is not present?

Часть 2. Настройка основных параметров коммутатора

Шаг 1. Присвойте коммутатору имя.

Для настройки параметров коммутатора, возможно, потребуется переключаться между режимами настройки. Обратите внимание, как изменяется командная строка при переходе между режимами командной строки коммутатора.

```
Switch# configure terminal
```

```
Switch(config)# hostname S1
```

```
S1(config)# exit
```

S1#

Шаг 2. Обеспечьте безопасный доступ к консоли.

Для безопасного доступа к консоли перейдите в режим config-line и установите для консоли пароль letmein.

S1# configure terminal

Enter configuration commands, one per line. End with CNTL/Z.

S1(config)# line console 0

S1(config-line)# password letmein

S1(config-line)# login

S1(config-line)# exit

S1(config)# exit

%SYS-5-CONFIG_I: Configured from console by console

S1#

Для чего нужна команда login?

Шаг 3. Убедитесь, что доступ к консоли защищен.

Выйдите из привилегированного режима, чтобы убедиться, что для консольного порта установлен пароль.

S1# exit

Switch con0 is now available

Press RETURN to get started.

User Access Verification

Password:

S1>

Примечание. Если коммутатор не выводит запрос на ввод пароля, значит, вы не настроили параметр login в шаге 2.

Шаг 4. Защитите доступ к привилегированному режиму.

Установите для enable пароль c1\$c0. Этот пароль ограничивает доступ к привилегированному режиму.

Примечание. Символ 0 в c1\$c0 — это ноль, а не заглавная буква «О». Настройка пароля будет оценена как выполненная успешно только после того как вы зашифруете его на шаге 8.

```
S1> enable
```

```
S1# configure terminal
```

```
S1(config)# enable password c1$c0
```

```
S1(config)# exit
```

```
%SYS-5-CONFIG_I: Configured from console by console
```

```
S1#
```

Шаг 5. Убедитесь, что доступ к привилегированному режиму защищен.

А. Введите команду `exit` еще раз, чтобы выйти из коммутатора.

Б. Нажмите <Enter>, после чего вам будет предложено ввести пароль.

```
User Access Verification
```

```
Password:
```

В. Первый пароль — это пароль для консоли, который был задан для `line con 0`. Введите этот пароль, чтобы вернуться в пользовательский режим EXEC.

Г. Введите команду для доступа к привилегированному режиму.

Д. Введите второй пароль, который был задан для ограничения доступа к привилегированному режиму EXEC.

Е. Проверьте конфигурацию, изучив содержимое файла `running-configuration`:

```
S1# show running-config
```

Обратите внимание, что пароли для консоли и привилегированного режима отображаются в виде обычного текста. Это может быть небезопасно, так как пароль может увидеть любой находящийся рядом человек.

Шаг 6. Настройте зашифрованный пароль для защиты доступа к привилегированному режиму.

Пароль `enable password` нужно заменить на новый зашифрованный пароль с помощью команды `enable secret`. Установите для `enable secret` пароль `itsasecret`.

```
S1# config t
S1(config)# enable secret itsasecret
S1(config)# exit
S1#
```

Примечание. Пароль `enable secret` имеет приоритет перед паролем `enable password`. Если для коммутатора заданы оба пароля, для перехода в привилегированный режим EXEC нужно ввести пароль `enable secret`.

Шаг 7. Убедитесь, что в файл конфигурации добавлен пароль `enable secret`.

А. Введите команду `show running-config` еще раз, чтобы проверить новый пароль `enable secret`.

Примечание. Команду `show running-config` можно сократить до

```
S1# show run
```

Б. Что отображается при выводе пароля `enable secret`?

В. Почему пароль `enable secret` отображается не так, как задан?

Шаг 8. Зашифруйте пароли `enable` и `console`.

Как было видно в шаге 7, пароль `enable secret` зашифрован, а пароли `enable password` и `console` хранятся в виде обычного текста. Сейчас мы зашифруем эти открытые пароли с помощью команды `service password-encryption`.

```
S1# config t
S1(config)# service password-encryption
S1(config)# exit
```

Если установить на коммутаторе другие пароли, они будут храниться в файле конфигурации в виде обычного текста или в зашифрованном виде? Дайте пояснение.

Часть 3. Настройка баннера MOTD (сообщения дня)

Шаг 1. Настройте баннер MOTD (сообщения дня).

В набор команд Cisco IOS входит команда, позволяющая настроить сообщение, которое будут видеть все, кто входит в систему на коммутаторе. Это сообщение называется сообщением дня или баннером MOTD (message of the day). Текст баннера нужно заключить в двойные кавычки или использовать разделитель, отличный от любого символа в строке MOTD.

```
S1# config t
```

```
S1(config)# banner motd "This is a secure system. Authorized Access Only!"
```

```
S1(config)# exit
```

```
%SYS-5-CONFIG_I: Configured from console by console
```

```
S1#
```

1. Когда будет отображаться этот баннер?
2. Зачем на всех коммутаторах должен быть баннер MOTD?

Часть 4. Сохранение файлов конфигурации в NVRAM

Шаг 1. Проверьте правильность конфигурации с помощью команды `show run`.

Шаг 2. Сохраните файл конфигурации.

Вы завершили основную настройку коммутатора. Теперь выполните резервное копирование файла конфигурации в NVRAM и убедитесь, что внесенные изменения не были потеряны при перезагрузке системы или отключении питания.

```
S1# copy running-config startup-config
```

```
Destination filename [startup-config]?[Enter]
```

```
Building configuration...
```

```
[OK]
```

Какова самая короткая версия команды `copy running-config startup-config`?

Шаг 3. Изучите файл загрузочной конфигурации.

Какая команда отображает содержимое NVRAM?

Все ли внесенные изменения были записаны в файл?

Часть 5. Настройка коммутатора S2

Вы завершили настройку коммутатора S1. Теперь настройте коммутатор S2. Если вы не можете вспомнить команды, вернитесь к частям 1–4.

Настройте для коммутатора S2 следующие параметры.

- А. Имя устройства: S2.
- Б. Защитите доступ к консоли паролем letmein.
- В. Задайте пароль enable password c1\$c0 и пароль enable secret itsasecret.
- Г. Введите следующее сообщение для пользователей, выполняющих вход в систему на коммутаторе:
Authorized access only. Unauthorized access is prohibited and violators will be prosecuted to the full extent of the law.
- Д. Зашифруйте все открытые пароли.
- Е. Проверьте правильность конфигурации.
- Ж. Сохраните файл конфигурации, чтобы предотвратить его потерю в случае отключения питания коммутатора.

Задание

Задание 1. Проверка конфигурации коммутатора по умолчанию

Задание 2. Настройка основных параметров коммутатора

Задание 3. Настройка баннера MOTD (сообщения дня)

Задание 4. Сохранение файлов конфигурации в NVRAM

Задание 5. Настройка коммутатора S2

ПРАКТИЧЕСКАЯ РАБОТА №7. КОНФИГУРИРОВАНИЕ КОММУТАТОРА

Цель: научиться проводить настройку начальных параметров коммутатора

Теоретическое обоснование

Общие сведения

В этом упражнении вы осуществите базовую настройку коммутатора. Затем вам будет необходимо обеспечить безопасность доступа к интерфейсу командной строки (CLI) и портам консоли с помощью зашифрованных и текстовых паролей. Вы также научитесь настраивать сообщения для пользователей, выполняющих вход в систему коммутатора. Эти баннеры также предупреждают пользователей о том, что несанкционированный доступ запрещен.

Часть 1. Проверка конфигурации коммутатора по умолчанию

Шаг 1. Войдите в привилегированный режим EXEC.

Привилегированный режим EXEC дает доступ ко всем командам коммутатора. Но поскольку многие привилегированные команды задают

рабочие параметры, привилегированный доступ должен быть защищен паролем во избежание несанкционированного использования.

Набор команд привилегированного режима EXEC включает команды, которые доступны в пользовательском режиме EXEC, а также команду `configure`, открывающую доступ к остальным командным режимам.

А. Щелкните S1, а затем вкладку CLI. Нажмите клавишу ввода.

Б. Перейдите в привилегированный режим EXEC, выполнив команду `enable`.

```
Switch> enable
```

```
Switch#
```

Обратите внимание, что командная строка изменилась, отображая переключение в привилегированный режим EXEC.

Шаг 2. Изучите текущую конфигурацию коммутатора.

А. Введите команду `show running-config`.

```
Switch# show running-config
```

Б. Ответьте на следующие вопросы.

1. Сколько у коммутатора интерфейсов FastEthernet?
2. Сколько у коммутатора интерфейсов Gigabit Ethernet?
3. Каков диапазон значений, отображаемых в линиях `vtu`?
4. Какая команда отображает текущее содержимое энергонезависимого ОЗУ (NVRAM)?
5. Почему коммутатор отвечает сообщением `startup-config is not present`?

Часть 2. Настройка основных параметров коммутатора

Шаг 1. Присвойте коммутатору имя.

Для настройки параметров коммутатора, возможно, потребуется переключаться между режимами настройки. Обратите внимание, как изменяется командная строка при переходе между режимами командной строки коммутатора.

```
Switch# configure terminal
```

```
Switch(config)# hostname S1
```

```
S1(config)# exit
```

```
S1#
```

Шаг 2. Обеспечьте безопасный доступ к консоли.

Для безопасного доступа к консоли перейдите в режим config-line и установите для консоли пароль letmein.

```
S1# configure terminal
```

Enter configuration commands, one per line. End with CNTL/Z.

```
S1(config)# line console 0
```

```
S1(config-line)# password letmein
```

```
S1(config-line)# login
```

```
S1(config-line)# exit
```

```
S1(config)# exit
```

```
%SYS-5-CONFIG_I: Configured from console by console
```

```
S1#
```

Для чего нужна команда login?

Шаг 3. Убедитесь, что доступ к консоли защищен.

Выйдите из привилегированного режима, чтобы убедиться, что для консольного порта установлен пароль.

```
S1# exit
```

Switch con0 is now available

Press RETURN to get started.

User Access Verification

Password:

```
S1>
```

Примечание. Если коммутатор не выводит запрос на ввод пароля, значит, вы не настроили параметр login в шаге 2.

Шаг 4. Защитите доступ к привилегированному режиму.

Установите для enable пароль c1\$c0. Этот пароль ограничивает доступ к привилегированному режиму.

Примечание. Символ 0 в c1\$c0 — это ноль, а не заглавная буква «О». Настройка пароля будет оценена как выполненная успешно только после того как вы зашифруете его на шаге 8.

```
S1> enable
```

```
S1# configure terminal
```

```
S1(config)# enable password c1$c0
```

```
S1(config)# exit
```

```
%SYS-5-CONFIG_I: Configured from console by console
```

```
S1#
```

Шаг 5. Убедитесь, что доступ к привилегированному режиму защищен.

А. Введите команду exit еще раз, чтобы выйти из коммутатора.

Б. Нажмите <Enter>, после чего вам будет предложено ввести пароль.

User Access Verification

Password:

В. Первый пароль — это пароль для консоли, который был задан для line con 0. Введите этот пароль, чтобы вернуться в пользовательский режим EXEC.

Г. Введите команду для доступа к привилегированному режиму.

Д. Введите второй пароль, который был задан для ограничения доступа к привилегированному режиму EXEC.

Е. Проверьте конфигурацию, изучив содержимое файла running-configuration:

```
S1# show running-config
```

Обратите внимание, что пароли для консоли и привилегированного режима отображаются в виде обычного текста. Это может быть небезопасно, так как пароль может увидеть любой находящийся рядом человек.

Шаг 6. Настройте зашифрованный пароль для защиты доступа к привилегированному режиму.

Пароль enable password нужно заменить на новый зашифрованный пароль с помощью команды enable secret. Установите для enable secret пароль itsasecret.

```
S1# config t
S1(config)# enable secret itsasecret
S1(config)# exit
S1#
```

Примечание. Пароль enable secret имеет приоритет перед паролем enable password. Если для коммутатора заданы оба пароля, для перехода в привилегированный режим EXEC нужно ввести пароль enable secret.

Шаг 7. Убедитесь, что в файл конфигурации добавлен пароль enable secret.

А. Введите команду show running-config еще раз, чтобы проверить новый пароль enable secret.

Примечание. Команду show running-config можно сократить до

```
S1# show run
```

Б. Что отображается при выводе пароля enable secret?

В. Почему пароль enable secret отображается не так, как задан?

Шаг 8. Зашифруйте пароли enable и console.

Как было видно в шаге 7, пароль enable secret зашифрован, а пароли enable password и console хранятся в виде обычного текста. Сейчас мы зашифруем эти открытые пароли с помощью команды service password-encryption.

```
S1# config t
S1(config)# service password-encryption
S1(config)# exit
```

Если установить на коммутаторе другие пароли, они будут храниться в файле конфигурации в виде обычного текста или в зашифрованном виде? Дайте пояснение.

Часть 3. Настройка баннера MOTD (сообщения дня)

Шаг 1. Настройте баннер MOTD (сообщения дня).

В набор команд Cisco IOS входит команда, позволяющая настроить сообщение, которое будут видеть все, кто входит в систему на коммутаторе. Это сообщение называется сообщением дня или баннером MOTD (message of the day). Текст баннера нужно заключить в двойные кавычки или использовать разделитель, отличный от любого символа в строке MOTD.

```
S1# config t
```

```
S1(config)# banner motd "This is a secure system. Authorized Access Only!"
```

```
S1(config)# exit
```

```
%SYS-5-CONFIG_I: Configured from console by console
```

```
S1#
```

1. Когда будет отображаться этот баннер?
2. Зачем на всех коммутаторах должен быть баннер MOTD?

Часть 4. Сохранение файлов конфигурации в NVRAM

Шаг 1. Проверьте правильность конфигурации с помощью команды `show run`.

Шаг 2. Сохраните файл конфигурации.

Вы завершили основную настройку коммутатора. Теперь выполните резервное копирование файла конфигурации в NVRAM и убедитесь, что внесенные изменения не были потеряны при перезагрузке системы или отключении питания.

```
S1# copy running-config startup-config
```

```
Destination filename [startup-config]?[Enter]
```

```
Building configuration...
```

```
[OK]
```

Какова самая короткая версия команды `copy running-config startup-config`?

Шаг 3. Изучите файл загрузочной конфигурации.

Какая команда отображает содержимое NVRAM?

Все ли внесенные изменения были записаны в файл?

Часть 5. Настройка коммутатора S2

Вы завершили настройку коммутатора S1. Теперь настройте коммутатор S2. Если вы не можете вспомнить команды, вернитесь к частям 1–4.

Настройте для коммутатора S2 следующие параметры.

А. Имя устройства: S2.

Б. Защитите доступ к консоли паролем `letmein`.

В. Задайте пароль `enable` `password c1$c0` и пароль `enable` `secret itsasecret`.

Г. Введите следующее сообщение для пользователей, выполняющих вход в систему на коммутаторе:

`Authorized access only. Unauthorized access is prohibited and violators will be prosecuted to the full extent of the law.`

Д. Зашифруйте все открытые пароли.

Е. Проверьте правильность конфигурации.

Ж. Сохраните файл конфигурации, чтобы предотвратить его потерю в случае отключения питания коммутатора.

Задание

Задание 1. Проверка конфигурации коммутатора по умолчанию

Задание 2. Настройка основных параметров коммутатора

Задание 3. Настройка баннера MOTD (сообщения дня)

Задание 4. Сохранение файлов конфигурации в NVRAM

Задание 5. Настройка коммутатора S2

ПРАКТИЧЕСКАЯ РАБОТА №8. ГИПЕРТЕКСТОВЫЕ ТЕХНОЛОГИИ. РАЗРАБОТКА WEB-САЙТА СТУДЕНТА

Цель работы: сформировать навыки разработки Web-сайта с применением HTML и CSS

Задание 1. Разобраться с понятиями дизайн и web-дизайн.

Ниже приводится подборка материала, в той или иной мере отвечающей на поставленный вопрос. Прочтите и составьте свое представление о предмете нашей лабораторной работы.

Что такое дизайн

ДИЗАЙН (от англ. design замысел, проект, чертеж, рисунок), термин, обозначающий различные виды проектировочной деятельности, имеющей целью формирование эстетических и функциональных качеств предметной среды. В узком смысле дизайн – художественное конструирование.

Ну а «художественное конструирование» – это «вид художественной деятельности, проектирование промышленных изделий, обладающих эстетическими свойствами».

Словарь Ожегова пишет о дизайне вот что:

ДИЗАЙН, а, м. Конструирование вещей, машин, интерьеров, основанное на принципах сочетания удобства, экономичности и красоты.

Артемий Лебедев говорит о дизайне следующее:

«... дизайн, это образ жизни и мысли – дизайнер, вопреки превратному представлению большинства, не только пиксели по экрану гоняет и не только мышью водит по столу. Он делает из хаоса порядок. Он владеет умами. Он создает настроения...»

Все правильно. Но не конкретно. На DeusExMachina есть даже отдельная статья, которая так и называется – «Что такое дизайн». Очень правильная статья.

Можно поспрашивать поисковые машины – что же такое дизайн.

Миша Галушко говорит:

Дизайн – это круглая сковородка.

И он прав, но об этом – ниже.

Итак, что же все-таки такое этот «дизайн?»...

«Дизайн – это умение рисовать такие загогулины, чтобы пользователь, зайдя на сайт, обязательно воскликнул: «Wow! Этот чувак явно крут в Фотошопе!»»

Кто-то не совсем правильно, верно? Поэтому многие пользуются для себя другим определением:

«Дизайн – это умение рисовать такие загогулины, чтобы пользователь, зайдя на сайт, обязательно воскликнул: «Wow! У этого чувака явно нет проблем с воображением, а скорее наоборот!»»

Зачем они это делают – я опишу позже, а пока – моя формулировка, что же такое дизайн.

Дизайн – конструирование вещей таким образом, чтобы ими было удобно, легко и приятно пользоваться по назначению и (практически) невозможно не по назначению.

(C) Spectator. И можете меня цитировать!

А теперь – буду обосновывать каждое слово. Чтобы не придрались. «Конструирование». Можно использовать «создание», «проектировка» и другие синонимы. То, что дизайн – это создание, думаю, объяснять не надо. Слово «вещь» здесь играет роль собирательного понятия. «Вещью» может быть и сайт, и молоток. Кому не нравится – может использовать слово «сущность», например. То есть дизайн – это создание сущностей.

Теперь надо описать характеристику создания сущностей, ибо не каждый процесс создания можно назвать дизайном. Прежде всего, у дизайнера есть цель. Четкая, прагматичная цель: сделать так, чтобы вещь *работала по назначению*, то есть выделить *существенные для работы признаки*. Выделить их таким образом, чтобы они были очевидны и «принуждали» пользователя поступать именно так, а не иначе.

К примеру, молоток. Выделяем признак, необходимый для качественной работы: способность забивать гвозди. Замечательно. Значит то место, каким молоток бьет по гвоздю, должно быть твердым и ровным. Логично? Вполне. Берем железную болванку. Гвозди ей уже забивать можно, но все еще не совсем удобно. Добавляем рычаг – рукоятку. Так значительно лучше. Минимум выполнен. А вот все, что идет за минимум и направлено на дальнейшее повышение удобства пользования – как раз то самое, что отличает хороший дизайн от плохого. Можно, например, поместить на рукоятку в некоторых местах резиновые полоски, чтобы рука не соскальзывала. Это будет дизайном. Можно сделать бьющую часть молотка из золота. Это дизайном не будет. Не потому, что золота жалко, а потому что золото – металл мягкий и такой молоток долго не продержится.

Тем не менее, я написал «удобно, легко и приятно пользоваться». В понятие «приятности» входит в том числе и эстетическая красота. Согласитесь, просто приятней держать в руках молоток, металлические части которого, например, хромированы, а не просто являются черными железками. На такой молоток приятней смотреть и другим не стыдно показать.

«...формирование эстетических и функциональных качеств предметной среды» – это как раз и есть то, что я перечислил. Формирование функциональных качеств – это выделение значимых признаков, необходимых для работы предмета, (удобство и легкость пользования) а формирование эстетических качеств – это создание «приятности» пользования предметом.

«...и (практически) невозможно не по назначению» – здесь я имел в виду «понятный» дизайн. Кнопки должны возжелеть, чтобы на них нажали. Они должны всем своим видом показывать, что они – кнопки. (Смотри пример ниже). Нормальный текст должен всем говорить: «я – нормальный текст! меня нужно читать!», а ссылки – «мы – ссылки! видите, как мы выделяемся? это неспроста!». Click here, как говорится. Идеальный дизайн – это такой дизайн, при котором вещь, попав к вам в руки, «сама по себе» занимает удобное

положение и вы уже «знаете» – чисто на уровне ощущений – что с этим предметом делать.

Идеальный дизайн – это такой дизайн, при котором вещь, попав к вам в руки, «сама по себе» занимает удобное положение и вы уже «знаете» – чисто на уровне ощущений – что с этим предметом делать.

Действительно, я как-то не совсем четко это обозначил. Исправляюсь: идеальный дизайн – это максимальное удобство использования плюс определенная доля эстетики, то есть «когда сущность «выглядит» как нечто уникальное и особенное», **НО!** только когда это не идет в ущерб удобству.

Что же такое web-дизайн?

Web-дизайн включает в себя - визуальный дизайн (вообще), дизайн представления информации на экране (специфика web-дизайна). Именно поэтому на данном сайте помещен цикл статей в том числе и о визуальном дизайне вообще (ссылка).

Поиск в интернете слова web-design (webdesign, web design) дает нам ссылки на инструкции и книги по руководствам к различным программам. Та же картина наблюдается в разделах "Web Design" таких сайтов-монстров, как WebDeveloper и C-Net Builder.com. Путаница связана с тем, что в русском языке слово "дизайн" приобрело другой смысл, нежели оно имеет в английском. В строгом смысле слова дизайн (design) - это разработка, конструирование. Сочетание "machine design" - переводится как "конструирование станков", а не как их эстетическое внешнее оформление.

Таким образом web-дизайнер в зарубежном понимании - это не кто иной как web-разработчик - верстальщик (web-master/web-coder), специалист по информационной архитектуре (web-developer), программист (web-programmer), администратор (system administrator) в одном лице.

В русском языке дизайн ближе к эстетике, изобразительному или оформительскому искусству - в таком понимании web-дизайнер это в первую очередь художник. То есть, то, что в английском языке соответствует слову web-artist, web graphical designer.

Из-за этого - люди, дающие в отечественных газетах объявления о том, что они являются веб-дизайнерами - на самом деле берут на себя непосильное обязательство web-разработчика, являясь всего лишь web-художниками (владея графическими пакетами типа Photoshop/Illustrator/3dstudio).

Из-за этого компании, дающие объявления о том, что им нужен web-дизайнер, на самом деле ищут не web-разработчика, а web-художника (как правило с художественным образованием), который знает, как сделать красиво вообще (с помощью все тех же Photoshop/Illustrator/3dstudio), но вовсе не обязан уметь это делать своими руками.

Из-за этого зарубежная книга по web-дизайну - это в большей части техническая информация о том, как верстать таблицы, как создавать навигацию, как оптимизировать изображения.

Из-за этого отечественная книга по web-дизайну (единственная на этот день - книга Кирсанова - это книга посвященная во многом визуальному дизайну.

(Обратите внимание на рецензию к книге Кирсанова. Вот цитата: "...[другие] авторы постоянно забывают (хочется надеяться, не намеренно) о том, что дизайн - будь он "веб", промышленный, художественный, садовый или интерьерный - прежде всего искусство, требующее наличия способностей, художественного вкуса, а также хотя бы минимальной теоретической подготовки у того, кто гордо именуется "дизайнером". - то есть рецензент как и ожидалось считает, что дизайн это прежде всего - эстетика.)

Сравните содержание книг Дмитрия Кирсанова и Якоба Нильсена. Обе они называются Web-дизайн. Но каждый из авторов пишет в своих книгах совершенно о разных вещах: Веб-дизайн: книга Якоба Нильсена и Веб-дизайн: книга Дмитрия Кирсанова

Из-за этого в Интернет, все что же касается художественной визуальной части веб-страниц, надо искать не в разделах **web-дизайн**, а в разделах **visual design** - именно там вы найдете все о сочетаемости цветов, о балансе композиции, контрасте, использовании шрифтов, создании коллажей, придумывании метафор и прочее.

Поэтому, если, скажем, проводится конкурс на лучший дизайн сайта, или идет обсуждения дизайна сайта, то оцениваться должна не навороченность графики и оригинальность рюшечек и менюшек, а то, насколько дизайнеру удалось создать удобный, работающий сайт. В WWW визуальность - средство, цель - удобство и эффективность.

Забывая об этом, некоторые веб-дизайнеры на своих личных страницах воруют такие вещи - которые основаны лишь на визуальном дизайне и строго говоря к веб-дизайну они имеют мало отношения, оно и понятно - личные страницы вряд ли ставят задачу представления информации - скорее задачу эксперимента, опыта, демонстрации, выпендрёжа наконец.

Чрезвычайное увлечение только визуальным дизайном может привести к плачевным результатам - делая сайт уже не для себя, а на заказ, web-художник (называющий себя web-дизайнером) может настолько сильно увлечься эстетическими наворотами и рюшечками, что совершенно забудет о другой стороне web-дизайна - навигации, аккуратности ссылок, совместимости с разными браузерами, легкости кода и изображений, удобстве администрирования сайта и еще о сотне всяческих мелочей.

С приходом Flash увеличился шанс появления на свет плохих сайтов. Вместо того, чтобы думать об удобстве пользователя - web-художник думает о том, чтобы такое наворотить с анимацией или со шрифтами кнопками и чтобы все летало и свистело и приводило пользователя в состояние транса. Получится так, что с эстетической точки зрения все будет прекрасно, но с практической - с сайтом просто невозможно будет работать.

Эта грань - между красотой и удобством - очень тонка. Это давний спор между эстетами и прагматиками

Задание 2. Язык разметки гипертекста HTML.

В практическом руководстве по HTML даются основные характеристики этого языка, с примерами использования. При работе с гипертекстовыми документами, значительную помощь вам может оказать справочник по тегам.

Для работы с гипертекстом необходимо использовать любой текстовый редактор, например, Блокнот.

Ознакомившись с основными требованиями, тегами и способами форматирования создайте следующие страницы:

1. **Главная** (титульная страница вашего сайта, на ней будут ссылки на остальные страницы),
2. **Обо мне** (автобиография, т.е. где родились, где учились, что умеете лучше всего делать и т.п.),
3. **Мои друзья** (сведения о ваших школьных и университетских друзьях: ФИО, чем увлекаются, сколько лет знакомы и т.д.),
4. **Мои увлечения** (то, чем вы любите заниматься в свободное от учебы время, какие фильмы, музыка и произведения вам нравятся, описав при этом, почему вы сделали такой выбор и др.),
5. **Я и моя группа** (написать свои отзывы о каждом из одноклассников и разместив рядом их фотографию).

Поместите созданные файлы в каталог **Сайт Иванова ИИ (ФИО заменить на свое)**, не забывайте - ВСЕ ФАЙЛЫ ДОЛЖНЫ ИМЕТЬ ЛАТИНСКИЕ НАЗВАНИЯ! Например: *about.htm*, *friends.htm*, *hobby.htm* и *groop.htm*. Фотографии одноклассников должны находиться в том же каталоге.

Если при создании страниц возникают трудности при форматировании текста воспользуйтесь "*Руководством по HTML*", упомянутым выше, или, приведенным ниже примером. При необходимости вставить в документ символ воспользуйтесь таблицей спец. символов.

<p>...</p>

Элемент абзаца разделяет фрагменты текста вертикальным отступом. Он позволяет использовать только начальный тэг, так как следующий элемент **</p>** обозначает не только начало следующего абзаца, но и конец предыдущего. В тех случаях, когда по смыслу необходимо обозначить завершение абзаца, можно использовать и конечный тэг. Вместе с элементом абзаца можно использовать и атрибут выравнивания **align**:

- **<p align="left">** – выравнивание по левому краю;
- **<p align="right">** – выравнивание по правому краю;
- **<p align="center">** – выравнивание по центру;
- **<p align="justify">** – выравнивание по ширине. Выравнивание по ширине – опасный параграф. Он не работает в старых версиях браузеров (т.е. текст выравнивается в таком случае по левому краю документа).

Пример:

```
<HTML>
<HEAD>
<TITLE>    Использование различных типов выравнивания    </title>
</head>
<BODY>
<P align="justify"> Элемент абзаца разделяет фрагменты текста вертикальным
отступом. Вместе с элементом абзаца можно использовать и атрибут
выравнивания align: </p>
<P align="left"> aligns ="left"—выравнивание по левому краю; </p>
<P align="center"> align="center" — выравнивание по центру; </p>
<P align="right"> align="right"—выравнивание по правому краю; </p>
<P align="justify"> align="justify" – выравнивание по ширине. (раз, два, три,
четыре ... раз, два, три, четыре ...раз, два, три, четыре ...раз, два, три, четыре
...раз, два, три, четыре ...раз, два, три, четыре ...)</p>
</body>
</html>
```


Элемент, обеспечивающий принудительный переход на новую строку. Он имеет только начальный тэг. В месте его размещения строка заканчивается, а оставшийся текст печатается с новой строки.

Пример:

```
<HTML>
<HEAD>
<TITLE>    Принудительный перенос строки    </title>
</head>
<BODY>
<HTML>
Вдруг как в сказке скрипнула дверь,<br>
Все мне ясно стало теперь,<br>
Сколько лет я спорил с судьбой,<br>
Ради этой встречи с тобой...<br>
</body>
</html>
```

Если таким образом расставить элемент
 в этом стихотворении, то в экране браузера мы увидим стандартным образом написанное четверостишие на четырех строках.

<NOBR>...</nobr>

Этот элемент по своему действию является прямой противоположностью предыдущему. Текст, заключенный между его тегам, будет выведен в одну строку. Длинная строка не уместится на экране, и для ее просмотра придется использовать горизонтальную полосу прокрутки.

<pre>...</pre>

Элемент для обозначения текста, отформатированного заранее (*preformatted*). Подразумевается, что текст будет выведен в том виде, в каком

он был подготовлен пользователем. 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. Во всех других случаях браузер игнорирует эти символы. Возможен и обратный эффект: если пользователь введет текст как одну длинную строку, то она не будет разорвана браузером, а уйдет за границы окна программы. В этом смысле элемент *PRE* работает так же, как элемент *NOBR*. По умолчанию для отформатированного заранее текста выбирается моноширинный шрифт. Этот элемент удобно использовать для демонстрации листингов программ или для вывода текстовых документов, переформатирование которых может привести к искажению их смысла.

`<BLOCKQUOTE> ... </blockquote>`

Текст, заключенный между метками `<BLOCKQUOTE>` и `</blockquote>`, выводится браузером на экран с увеличенным левым полем (с отступом).

`<CENTER>...</center>`

Элемент для центрирования текста, а точнее, любого содержимого. Не является общеупотребительным. В тех случаях, когда это возможно, вместо него используют атрибут *align = "center"*.

`<DIV>...</div>`

Элемент, похожий на предыдущий. Он позволяет выравнивать содержимое по левому краю, по центру или по правому краю. Для этого стартовый тег должен содержать соответствующий атрибут:

- *align = "left"*
- *align = "center"*
- *align = "right"*

Форматирование шрифта HTML допускает два подхода к шрифтовому выделению фрагментов текста – логическое и физическое форматирование.

В группу тегов логического форматирования входят теги, отображающие на экране монитора элементы документа, таким образом, как установлено по умолчанию в спецификации языка разметки HTML. Переопределить их параметры или свойства нельзя, за исключением ситуаций использования стилевых шаблонов CSS и обособления тегами физического форматирования. Результат действия разных тегов логического форматирования визуально может совпадать, ибо основное их предназначение заключается в логическом выделении отдельных элементов HTML.

Теги физического форматирования позволяют разработчику визуально изменять вид текста, делая его жирным или наклонным, варьируя его параметры и значения. Под физическом стилем принято понимать прямое указание браузеру на модификацию текущего шрифта.

`<B>...</b>`

Выделение текста полужирным шрифтом (от слова *Bold*), - очень популярный элемент. Использование полужирного шрифта – прием, позаимствованный из текстовых редакторов. Т.е. текст находящийся внутри тега будет **жирным**.

<I>...</i>

Выделение текста курсивом (от слова *italic*). Т.е. текст, находящийся внутри тега *под наклоном*

<TT>...</tt>

Элемент, обозначающий текст телетайпа (*teletype*). Его особенность заключается в использовании моноширинного шрифта, имитирующего пишущую машинку, то есть имеющего фиксированную ширину символа.

<STRIKE>...</strike>

Элемент, создающий перечеркнутое начертание текста. В настоящее время его заменяют тегом: **<DEL> </del>**.

<U>...</u>

Подчеркнутое начертание текста.

_{...}

Элемент, создающий эффект нижнего индекса (*subscript*).

^{...}

Элемент, создающий эффект верхнего индекса (*superscript*).

Размер шрифта

<BASEFONT>

Элемент, определяющий базовый (основной) размер шрифта. Внутри элемента необходимо указать атрибут:

size = Базовый размер шрифта

Величина атрибута может лежать в пределах от 1 до 7. По умолчанию используется величина 3. Установка, выполняемая этим элементом, имеет значение для элемента *FONT* (см. ниже), который позволяет задавать относительный размер шрифта.

...

Определение типа, размера и цвета шрифта. Все эти характеристики определяются при помощи соответствующих атрибутов. Абсолютный размер шрифта задается при помощи атрибута size (размер):

size=Абсолютный размер шрифта

Этот атрибут может принимать значения от 1 до 7. В таблице показано несколько образцов надписей, выполненных шрифтами разного размера.

Пример

** Тыше едешь - дальше будешь **

Пример:

<HTML>

<HEAD>

<TITLE> Образцы надписей, выполненных шрифтами разного размера </title>

</head>

<BODY>

**
 SIZE=1 Образец написания **

**
 SIZE=2 Образец написания **

**
 SIZE=3 Образец написания **

Правда, есть одна проблема. Web-страницы просматривает множество людей, и нет гарантии, что у каждого из них окажется нужный шрифт. Если в системе не установлен шрифт точно с таким же названием, то браузер использует свой стандартный. Он имеет два назначенных по умолчанию шрифта: один пропорциональный, другой моноширинный.

Все эти атрибуты могут быть использованы совместно внутри тэга `<font>`.

Пример

`<FONT face="Arial" size=5 color="blue" >` Это шрифт arial размером 5, цвет синий `</font>`

Элемент **FONT** может с успехом заменять элементы заголовка *H1...H6*. Для последних, например, не предусмотрена возможность указания цвета букв.

Чтобы заголовок, созданный на основе элемента *FONT*, хорошо смотрелся, этот элемент необходимо комбинировать с другими: *CENTER*, *B*, *I*, *P* и т. д.

`<BIG>...</big>`

Этот тег используется, если необходимо выделить часть текста небольшим увеличением размера шрифта относительно остальных слов.

`<SMALL>...</small>`

Этот тег используется, если необходимо выделить часть текста небольшим уменьшением размера шрифта относительно остальных слов.

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.

`<H1>...</h1>`

Элемент заголовка. Существует шесть уровней заголовков, которые обозначаются *H1*. . *H6*. Заголовок уровня 1 самый крупный, а уровень 6 обеспечивает самый маленький заголовок. Для заголовков можно использовать атрибут, задающий выравнивание влево, по центру или вправо:

align	=	"left"
align = "right"	=	"center"

Пример:

`<HTML>`

`<HEAD>`

`<TITLE>Использование`

заголовков`</title>`

`</head>`

`<BODY>`

`<H1>Заголовок 1</H1>`

```
<H2>Заголовок 2</H2>
<H3>Заголовок 3</H3>
<H4>Заголовок 4</H4>
<H5>Заголовок 5</H5>
<H6>Заголовок 6</H6>
</body>
</html>
```

<ACRONYM>...</acronym>

Данный элемент реализуется через параметр *TITLE* и отображается в браузере при наведении курсора на слово-аббревиатуру

Пример:

```
<HTML>
<HEAD>
<TITLE>          Расшифровка          аббревиатур          </title>
</head>
<BODY>
Термин
<B><ACRONYM TITLE="HyperText Markup Language"> HTML </acronym>
</b>- означает "язык маркировки гипертекстов" и включает в себя различные
способы оформления гипертекстовых документов, дизайн, гипертекстовые
редакторы,          браузеры          и          многое          другое.
</body>
</html>
```

** и <DFN> </dfn>**

Элементы, означающие выразительность (от английского *emphasis* - акцент) фрагмента текста и определение чего-либо (*definition*). Оба элемента аналогичны по своему действию элементу *<i>*, то есть, в большинстве случаев, позволяют выделить текст курсивом. Казалось бы, два последних элемента - лишние. С точки зрения дизайна документа это так. Они могут пригодиться только для того, чтобы единообразно выделить одинаковые по назначению (или смыслу) фрагменты текста, находящиеся в разных частях документа или даже на разных страницах. Разработчик в этом случае не может точно знать, какой именно шрифт будет использован: это определяется каждым браузером по-своему. Но он может быть уверен, что все фрагменты текста будут отформатированы одинаково. В языке можно найти еще несколько элементов, которым можно дать такую же характеристику.

<CITE>...</cite>

Предполагается, что этот элемент может быть использован для форматирования цитат, высказываний и ссылок в обычном понимании этого слова. Текст, расположенный внутри него, выводится по умолчанию курсивом. Визуально аналогичен тегам ** и *<I>*

<ADDRESS>...</address>

Подобно элементу *CITE*, данный элемент отличается только предусмотренным содержанием. Он также обеспечивает форматирование курсивом.

...

От английского *strong emphasis* - сильный акцент.

Элемент, отвечающий за выделение текста. Обычно его применение равносильно использованию элемента для выделения полужирным: ****.

<KBD> ... </kbd>

От английского *keyboard* - клавиатура.

Этот элемент предназначен для указания текста, который пользователь должен ввести с клавиатуры (keyboard). Можно рассчитывать, что текст, выделенный с помощью этого элемента, будет выведен моноширинным шрифтом в полужирном начертании.

<CODE> </code>, <SAMP> </samp> и <VAR> </var>

Элементы, предназначенные для вывода фрагментов программ.:

· **<CODE>** ... **</code>**

Этот элемент предназначен для визуального выделения небольших фрагментов программного кода. Код отображается моноширинным шрифтом.

· **<SAMP> ... </samp>** - от английского *sample* - образец.

Рекомендуется использовать для демонстрации образцов сообщений, выводимых на экран программами. SAMP предполагается задействовать при иллюстрации примеров (sample) вывода данных на экран.

· **<VAR> ... </var>** - от английского *variable* - переменная.

Этот элемент был создан для выделения переменных (variables). Как правило, все эти элементы обеспечивают вывод информации с использованием моноширинного шрифта.

<PLAINTEXT>...</plaintext>

Этот элемент предназначен для создания текста с конструкциями HTML, которые должны восприниматься именно как текст. Все теги, заключенные в *PLAINTEXT*, воспринимаются браузером только как произвольные символы. Элемент удобно использовать для обсуждения вопросов, связанных с использованием HTML. Горизонтальные линии

Горизонтальная линия (*horizontal rule*) или, на типографский манер, горизонтальные линейки - очень часто используемый элемент. Во-первых, потому, что с его помощью очень удобно делить страницу на части, использовать для отделения одного блока текста от другого. Небольшой по размеру текст, сверху и снизу которого располагаются горизонтальные линии, привлекает больше, чем обычно, внимание читателя. Во-вторых, выбор таких элементов оформления у автора страницы очень небольшой. Действительно, в HTML практически отсутствуют подобные конструкции, а вот для горизонтальной линии почему-то было сделано исключение. Правда, несмотря на некоторую скупость языка в этой области, можно придумать немало стандартных графических образов, которые разнообразили бы вид страниц.

Если нужна простая, без затей линия, можно воспользоваться тегом **<HR>**.

Пример создания горизонтальных линий с помощью тега *HR*

Элемент не имеет конечного тега, но допускает ряд атрибутов:

1. `<HR align="right">`-выравнивание влево, по центру, вправо, по ширине: align = "left"; align = "center"; align = "right"; align = "justify"
2. `<HR width="30%">`-длина линии в процентах и пикселях
width = Длина в пикселях width = Длина в процентах % от ширины экрана
3. `<HR size="6">`-Толщина линии
4. `<HR NoShade>`-Отмена объемности
5. `<HR color="red">`-цвет линии, только в IE

Пример:

`<HTML>`

`<HEAD>`

`<TITLE>` Горизонтальные линии `</title>`

`</head>`

`<BODY>`

Длина линии - 30% от ширины экрана. Использование выравнивания линии по правому краю, по центру и по левому краю

`<HR width="30%" align="right">`

`<HR width="30%" align="center">`

`<HR width="30%" align="left">`

`<BR>`

толщина линии - 2, длина -100% `<HR SIZE=2 WIDTH=100%>`

толщина линии - 4, длина -50% `<HR SIZE=4 WIDTH=50% color="#00E0A9">`

толщина линии - 8, длина -25% `<HR SIZE=8 WIDTH=25%>`

толщина линии - 16, длина -12% `<HR SIZE=16 WIDTH=12% color="blue">`

Линия, не содержащая объема `<HR NoShade>`

`</body>`

`</html>`

Escape последовательности (символьные объекты)

Некоторые символы, такие как "<" и ">" воспринимаются браузерами как начало и конец HTML-меток, поэтому должен существовать способ их выражения, как символьных данных внутри самого документа или в URL. В HTML это делается с помощью &-последовательностей (их еще называют символьными объектами или escape (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). Принято использовать следующие нотации (соответствия):

символ обычное имя символа HTML запись символа (escape последовательность)

<	символ "меньше чем", левая угловая скобка	<
>	символ "больше чем", правая угловая скобка	>
&	амперсанд	&
"	двойные кавычки	"

с знак копирайт ©
R знак зарегистрированной торговой марки ® непрерывный пробел

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.

Задание 3. Выбор цветовой палитры сайта.

Цветовая палитра является одним из самых важных элементов оформления Web сайта. Познакомьтесь с основными характеристиками цвета: {яркость, оттенок, насыщенность} {яркость} {насыщенность} {оттенок}.

ТЕПЛЫЕ И ХОЛОДНЫЕ ЦВЕТА

Такая характеристика цвета как теплота во многом определяет воздействия цвета на человека. Собственно, даже для непосвященного человека будет несложным отличить теплый цвет от холодного. Это как раз тот случай, когда оценка зависит от простой эмоциональной оценки. Теплые цвета кажутся близкими, добрыми, внушающими доверие, в то время как холодные как бы находятся на расстоянии, независимы. Если мы обратим свой взгляд на цветовой круг, то увидим, что к теплым можно отнести оранжевые, красные и желтые оттенки. К холодным же мы причислим оттенки зеленого и синего. Следует, правда, обратить внимание на то, что существует область цветового круга, оттенки которого могут быть восприняты и как теплые и как холодные.

Как видно на следующем примере цвет слева является холодным, цвет справа скорее теплый, а оттенок посередине может показаться одновременно и теплым и холодным.

Посмотрите также на еще несколько вариантов определения цвета, отталкиваясь от его теплоты.

Оттенки синего цвета, начиная с теплых, имеющих синюю основу, заканчивая цветами с холодной зеленой основой

Аналогично цвет слева имеет желтую основу, а поэтому воспринимается как теплый. В это же время цвета справа имеют синюю основу и считаются холодными

На этих трех параметрах основана весьма удобная в обращении цветовая модель HSB (hue, saturation, brightness), которая входит в большинство графических пакетов. Эта модель не обеспечивает такой степени точности, как RGB, зато делает процесс подбора цвета более наглядным.

Оттенок - какой то определенный цвет, красный или синий, например.

Насыщенность - степень чистоты цвета. На приведенном примере справа мы имеем дело с оттенками, где высоко присутствие красного цвета, слева же мы видим приглушенные вариации красного, где чувствуется присутствие серого.

Яркость - степень присутствия в цвете черного или белого. Слева голубой цвет близок к белому, поэтому он "яркий". Слева цвет близок к черному, поэтому считается "темным".

Яркость

Яркость цвета - важный элемент любого дизайна, влияющий на восприятие в целом, удобочитаемость текста, эмоциональное восприятие. Во многом именно яркость определяет пространственное восприятие изображения.

В качестве примера рассмотрим изображение справа. Так, **контраст** цветов по яркости определяет положение объектов в пространстве, отделяет артишок на изображении от фона, а также разграничивает листья растения. В это же время **градации** яркости показывают контуры форм объектов изображения.

Как определить яркость цвета?

В то же самое время достаточно сложно сходу и главное точно определить степень яркости того или иного оттенка, сравнить два цвета по яркости. Например, можете ли вы с уверенностью сказать, какой цвет в данных парах более яркий?

Уверен, вы сделаете определенный выбор. На самом же деле обе цветовые пары имеют одинаковые показатели яркости.

Существует удобный способ для определения яркости цвета. Как мы уже знаем, именно степень присутствия белого или черного и определяет яркость. По этому для определения степени яркости вполне логично сопоставить следующую градацию синего цвета с его черно-белым аналогом.

Для подбора цвета своей страницы вы можете воспользоваться редактором `cveta`.

Используя таблицы цветов, выберите палитру из 18 рядом лежащих цветов, которые вы будете использовать для своего сайта. Оформите отдельную страничку "***palitra.htm***" (*Набор палитр*) с этим набором и разместите ее в разделе "**Учебная работа**". Для помощи можете воспользоваться **Набором палитр для WEB-дизайна**. Выбрав подходящее сочетание цветов можете сделать скриншот и, обработав изображение при помощи Photoshop, подвязать его к своей странице.

Предлагаем вашему вниманию палитру цветов, рекомендуемых для дизайна на WWW.

Когда вы создаете изображения для публикации их на Интернетe, главной проблемой является правильная передача цветов на разных типах компьютеров, мониторах и броузерах. Когда броузер не в состоянии правильным образом передать тот или иной цвет, он подбирает другой цвет или смешивает несколько соседних цветов (dithering). Иногда цвет может быть заменен на что-то совершенно неподходящее.

При использовании цветов из нашей палитры вы можете быть уверены в том, что каждый цвет будет передан правильным образом и без искажений. Любой из 216 цветов нашей палитры может быть использован для графики, текста и фонов.

Палитра состоит из всех возможных сочетаний 6 оттенков красного, зеленого и синего. Эти градации (в RGB) - 0, 51, 102, 153, 204, 255.

КАК ПОЛЬЗОВАТЬСЯ ПАЛИТРОЙ

Каждый цвет помечен номером по каталогу Pantone Colorweb. Под каждым цветом даны два значения - RGB (для создания цвета в графическом редакторе) и HEX (для обозначения цвета в HTML). Если вы хотите добиться максимального эффекта, не используйте форматов с потерей качества (напр., JPEG), лучше всего подойдет GIF.

255.25 5.204	255.25 5.153	255.25 5.102	255.25 5.51	255.2 55.0	204.2 04.0
FFFFC	FFFF99	FFFF66	FFFF3	FFFF	CCC
C			3	00	C00

255.204 .102	255.2 04.0	255.20 4.51	204.1 53.0	204.15 3.51	153.1 02.0
FFCC66	FFCC	FFCC3	CC99	CC993	99660
00		3	00	3	0

255.1 53.0	255.15 3.51	204.153 102	204.1 02.0	153.10 2.51	102.5 1.0
FF990	FF9933	CC9966	CC66	996633	6633
0			00		00

255.204 153	255.153 102	255.10 2.0	204.10 2.51	153.5 1.0	102. 0.0
FFCC99	FF9966	FF660	CC663	9933	660
		0	3	00	000

255.102 51	204.51 .0	255.51 .0	255.0 0	204.0 0	153.0 .0
FF6633	CC330	FF330	FF00	CC00	9900
	0	0	00	00	00

255.204	255.153	255.102	255.5	255.0	204.0
.204	.153	.102	1.51	.51	.51
FFCCC	FF9999	FF6666	FF333	FF00	CC0
C			3	33	033

204.153.	204.102.	204.51	153.51	153.0	51.0.
153	102	.51	.51	.51	0
CC9999	CC6666	CC333	99333	9900	3300
		3	3	33	00

255.102	255.51	255.0.	204.51	153.102	102.5
.153	.102	102	.102	.102	1.51
FF6699	FF336	FF00	CC336	996666	66333
	6	66	6		3

255.153.	255.51.	255.0.	204.0.	153.51.	102.0
204	153	153	102	102	.51
FF99CC	FF3399	FF009	CC00	993366	6600
		9	66		33

255.102	255.0.	255.51	204.102	204.0.	153.0.
.204	204	.204	.153	153	102
FF66C	FF00	FF33C	CC6699	CC00	99006
C	CC	C		99	6

255.20	255.15	255.10	255.51	255.0	204.51
4.255	3.255	2.255	.255	.255	.153
FFCCF	FF99F	FF66F	FF33F	FF00	CC33
F	F	F	F	FF	99

204.153	204.102	204.0.	204.51	153.0.	153.51
.204	.204	204	.204	153	.153
CC99C	CC66C	CC00	CC33	99009	99339
C	C	CC	CC	9	9

204.102	204.51	204.0.	153.0.	153.102	102.0.
.255	.255	255	204	.153	102
CC66F	CC33F	CC00	9900	996699	66006
F	F	FF	CC		6

204.153	153.51	153.51	153.0.	102.0.	102.51
.255	.204	.255	255	153	.102
CC99F	9933C	9933F	9900F	66009	66336
F	C	F	F	9	6

153.102	153.102	102.0.	102.51.	102.51.	51.0
.204	.255	204	204	153	.51
9966CC	9966FF	6600	6633C	663399	330
		CC	C		033

204.204.	153.153.	102.51.	102.0.	55.0.	51.0.
255	255	255	255	153	102
CCCCF	9999FF	6633FF	6600F	3300	3300
F			F	99	66

153.15	102.10	102.10	102.10	51.51	51.51
3.204	2.255	2.204	2.153	.153	.102
9999C	6666FF	6666C	666699	3333	3333
C		C		99	66

51.51.2	51.0.2	51.0.2	51.51.2	0.0.1	0.0.1
55	55	04	04	53	02
3333FF	3300F	3300C	3333CC	0000	0000
	F	C		99	66

102.153.2	51.102.2	0.0.2	0.0.20	0.51.2	0.0.5
55	55	55	4	04	1
6699FF	3366FF	0000	0000	0033C	0000
		FF	CC	C	33



55	0.102.2	04	0.102.2	204	51.102.	55	0.51.2	53	0.51.1	02	0.51.1
F	0066F	C	0066C		3366CC	F	0033F	9	00339	6	00336

.255	153.204	.255	51.153	255	0.153.	.204	102.153	.153	51.102	153	0.102.
F	99CCF	F	3399F	FF	0099	C	6699C	9	33669	9	00669

255	102.204.	255	51.204.	255	0.204.	204	51.153.	204	0.153.	.51	0.51
	66CCFF	F	33CCF	FF	00CC	C	3399C	C	0099C	333	003

4.204	153.20	4.204	102.20	.153	51.153	3.153	102.15	.102	0.102	.102	51.102
C	99CCC	C	66CCC	9	33999		669999	66	0066	6	33666

5.255	204.25	5.255	153.25	5.255	102.25	.255	51.255	.255	0.255	.204	0.204
F	CCFFF		99FFFF		66FFFF	F	33FFF	FF	00FF	CC	00CC

.204	153.255	.204	102.255	.204	51.255	204	0.255.	.204	51.204	153	0.153.
C	99FFC	C	66FFC	C	33FFC	CC	00FF	CC	33CC	9	00999

153	102.204.	153	51.204.	153	0.204.	102	51.153.	102	0.153.	.51	0.102
-----	----------	-----	---------	-----	--------	-----	---------	-----	--------	-----	-------

153.2 55.0	153.25 5.51	153.204 .102	153.2 04.0	153.20 4.51	102.1 53.0
99FF0 0	99FF3 3	99CC66	99CC 00	99CC3 3	66990 0

204.255 .102	204.2 55.0	204.25 5.51	204.204 .153	102.10 2.51	51.5 1.0
CCFF66 00	CCFF 00	CCFF3 3	CCCC9 9	666633	333 300

204.204 .102	204.20 4.51	153.15 3.51	153.153 .102	153.1 53.0	102.1 02.0
CCCC6 6	CCCC 33	99996 6	999933	99990 0	66660 0

255.255 .255	204.204 .102	153.153 .153	102.102 .102	51.5 1.51	0.0. 0
FFFFFF C	CCCCC C	999999	666666	3333 33	000 000

Задание 4. Создание гиперссылок.

В результате выполненных лабораторных работ у вас должно получиться 6 html-страниц, которые необходимо объединить гиперссылками.

Отчетность по практическим работам

Необходимо написать отчет и подвесить в своем профиле в систему университета **Еcampus.ncfu.ru** (и/или отправить на адрес zamira.ncfu@gmail.com) детальное сообщение о проделанной работе, прикрыв архив с каталогом с созданными страницами.

В строке 'Тема' письма, указать номер лабораторной работы и ваше ФИО.

Письмо должно содержать приветствие, что выполнено - по пунктам и что не выполнено, подпись автора.

ПРАКТИЧЕСКАЯ РАБОТА №9. РАЗРАБОТКА КАСКАДНЫХ ТАБЛИЦ СТИЛЕЙ (CSS). ИСПОЛЬЗОВАНИЕ ТАБЛИЦ

В практической работе мы изучим и создадим свою каскадную таблицу стилей для web-сайта студента

Если Вы решили создать HTML-документ с разветвленной структурой, постарайтесь оформить все его страницы одинаково, оперируя в рамках одного стиля. применяйте повторяющиеся управляющие элементы, одинаковые интервалы и подобные шрифты.

Это относится и к регулярному обновлению Web-страниц. Чтобы добиться стилевого единства, можно создать ряд шаблонов, которые подлежат заполнению в текстовом редакторе. Временные затраты на разработку шаблонов быстро окупятся при больших объемах документов и частом обновлении содержимого.

Задание. Ознакомьтесь со способами создания и применения каскадных таблиц стилей.

Спецификация CSS (Cascading Style Sheets) позволяет остаться в рамках декларативного характера разметки страницы и дает полный контроль над формой представления элементов HTML-разметки. Изучите назначение, способы применения и синтаксис для CSS.

CSS (Cascading Style Sheets) - язык таблиц стилей, который позволяет прикреплять стиль (например, шрифты и цвет) к структурированным документам (например, документам HTML и приложениям XML).

Обычно CSS-стили используются для создания и изменения стиля элементов веб-страниц и пользовательских интерфейсов, написанных на языках HTML и XHTML, но также могут быть применены к любому виду XML-документа, в том числе XML, SVG и XUL.

Отделяя стиль представления документов от содержимого документов, CSS упрощает создание веб-страниц и обслуживание сайтов.

CSS поддерживает таблицы стилей для конкретных носителей, поэтому авторы могут адаптировать представление своих документов к визуальным браузерам, слуховым устройствам, принтерам, брайлевским устройствам, карманным устройствам и т.д.

Каскадные таблицы стилей описывают правила форматирования элементов с помощью свойств и допустимых значений этих свойств. Для каждого элемента можно использовать ограниченный набор свойств, остальные свойства не будут оказывать на него никакого влияния.

Объявление стиля состоит из двух частей: селектора и объявления. В HTML имена элементов нечувствительны к регистру, поэтому «h1» работает так же, как и «H1». Объявление состоит из двух частей: имя свойства (например, color) и значение свойства (grey). Селектор сообщает браузеру, какой именно элемент форматировать, а в блоке объявления (код в фигурных скобках) перечисляются формирующие команды - свойства и их значения.

Создайте CSS- файл style.css (образец прикреплен), внутри которого будет задан ВАШ стиль для страниц. А именно цвет и стиль шрифта, цвет и

стиль ссылок, а также стиль объектов и текста, помещенных между тегами, которые вы использовали при создании своих страниц (например цвет и положение списков, положение рисунков и т.д.).

Необходимо подключить данный файл ко всем своим страницам с помощью следующего кода:

```
<head>

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

Отчетность по практической работе

Необходимо написать отчет и подвесить в своем профиле в систему университета Ecampus.ncfu.ru (и/или отправить на адрес zamira.ncfu@gmail.com) детальное сообщение о проделанной работе, прикрепив архив с каталогом с созданными страницами, с уже подвязанной CSS.

В строке 'Тема' письма, указать номер практической работы и ваше ФИО.

Письмо должно содержать приветствие, что выполнено - по пунктам и что не выполнено, подпись автора.

Задание 3. Как устроены таблицы.

Самый простой механизм размещения материала сайта - это таблицы. Это самый распространенный и почти единственный инструмент позиционирования объектов веб-страницы. Обычно таблицу и используют исключительно для этих целей. Чаще всего с помощью таблиц составляют сложные композиции из отдельных рисунков.

Позиционирование. Таблицы

Самый простой механизм размещения материала сайта - это таблицы. Это самый распространенный и почти единственный инструмент позиционирования объектов веб-страницы. Обычно таблицу и используют исключительно для этих целей. Чаще всего с помощью таблиц составляют сложные композиции из отдельных рисунков.

Теперь поговорим о достоинствах и недостатках данного метода.

Самый, пожалуй, досадный недостаток в том, что величина полей таблицы не поддается точному контролю. Здесь приходят на помощь разные ухищрения - распорки, вставка рисунков где иногда они бывают не к месту. И еще, браузер не отобразит таблицу с ее содержимым до тех пор, пока не будут известны размеры всех полей таблицы. То есть, если в конце таблицы будет содержаться рисунок, а перед ним объемный текст, пользователь будет ждать загрузки всей таблицы до тех пор, пока не будет загружено начало последнего рисунка (в том случае если в теге `<IMG SRC=. .>` не определены атрибуты `width` и `height`), что может вызвать вполне обоснованное раздражение

пользователя. Поэтому в рисунках всегда указывайте атрибуты width и height в теге .

Когда вы используете таблицу исключительно для текста, при этом важное условие (посмотрите любую газету) чтобы все столбцы были одинаковой ширины, иначе отклонения будут восприниматься как дефект, так казалось бы нет ничего проще: в теге <TABLE> задаете атрибут width=100%, а в каждом теге <TD> - width=33% для трех столбцов. Но это не всегда срабатывает: что, если в каком-либо столбце содержится слишком длинное слово? При уменьшении размера окна размеры столбцов начнут меняться, причем не равномерно. Чтобы этого не произошло используют распорки из прозрачных GIF-ов с атрибутами width="фиксированное значение" и height=1. В таком случае при уменьшении окна появится горизонтальная прокрутка. Нежелательный элемент? А кто пользователя заставляет минимизировать окно? Также распорки применяют и для позиционирования по вертикали внутри столбца.

Если некоторые ячейки объединены, как в этом примере, - объединенные ячейки должны иметь фиксированный размер в сторону объединения, иными словами браузер "не знает" какие из соседних ячеек растягивать. Для этого в тегах <TD> задаются атрибуты width и height. При использовании объединения ячеек приходится фиксировать размеры в теге <TABLE>. Это не всегда удобно, так как появляется вероятность возникновения горизонтальной прокрутки, что часто не желательно, но что делать... Можно выйти из этой проблемы только пожалуй вложенными таблицами. Эта страница - немного иной случай, здесь изображено множество ячеек в 1 пиксел, залитых цветом, - так изображены линейки. Линейки, конечно можно изобразить используя соответствующие графические бэкграунды в каждой ячейке, но я умышленно пошел по более сложному пути и, кстати не совсем правильному. Поэтому могу дать единственный совет: не сращивайте ячейки атрибутами ROWSPAN и COLSPAN - это только добавит проблем, лучше используйте вложенные таблицы если уж никак не обойтись. Исключение может составить только позиционирование составных изображений.

Есть и еще не маловажная проблема: например на этой странице пользователь может менять размер шрифта в своем браузере, а также и сам шрифт, хотя я специально использовал тег <FONT FACE . . . , вот только стоит изменить размер шрифта, как сразу все изменится не в лучшую сторону.

Что можно еще сказать о таблицах? Вероятно этого краткого обзора будет и достаточно для того, чтобы выяснить особые моменты при проектировании веб-страниц с использованием таблиц. Синтаксис и спецификацию я не привожу, так как этот инструмент достаточно распространен, да и просто мы здесь не рассматриваем технических приемов, - имеются другие разделы.

Очень внимательно разберитесь с методами построения страницы. Для работы с исходным текстом этой страницы можно сохранить страницу в HTML-виде браузером.

Задание 2. Таблицы в примерах

Чтобы разобраться с технологией построения таблиц можете воспользоваться примерами построения таблиц различной конфигурации. При создании таблиц можно устанавливать распорки, удерживающие ее размеры.

Таблицы в примерах

Этот документ содержит примеры таблиц. Он охватывает большинство новых тэгов в таблицах. Впрочем он не обязательно показывает некоторые действительно творческие возможности, доступные в таблицах.

ОБЫЧНАЯ ТАБЛИЦА 3X2

A	B	C
D	E	F

```
<TABLE BORDER>
<TR>
  <TD>A</TD> <TD>B</TD> <TD>C</TD>
</TR>
<TR>
  <TD>D</TD> <TD>E</TD> <TD>F</TD>
</TR>
</TABLE>
```

ДВА ПРИМЕРА С ROWSPAN

Item 1	Item 2	Item 3
Item 4		Item 5

```
<TABLE BORDER>
<TR>
  <TD>Item 1</TD>
  <TD ROWSPAN=2>Item 2</TD>
  <TD>Item 3</TD>
</TR>
<TR>
  <TD>Item 4</TD> <TD>Item 5</TD>
</TR>
</TABLE>
```

Item 1	Item 2	Item 3	Item 4
	Item 5	Item 6	Item 7

```
<TABLE BORDER>
<TR>
  <TD ROWSPAN=2>Item 1</TD>
  <TD>Item 2</TD> <TD>Item 3</TD> <TD>Item 4</TD>
</TR>
<TR>
```

```

        <TD>Item 5</TD> <TD>Item 6</TD> <TD>Item 7</TD>
    </TR>
</TABLE>

```

ПРИМЕР С COLSPAN

Item 1	Item 2	
Item 3	Item 4	Item 5

```

<TABLE BORDER>
<TR>
    <TD>Item 1</TD>
    <TD COLSPAN=2>Item 2</TD>
</TR>
<TR>
    <TD>Item 3</TD> <TD>Item 4</TD> <TD>Item 5</TD>
</TR>
</TABLE>

```

ДЕМОНСТРАЦИЯ ЗАГОЛОВКОВ (<TH>)

Head1	Head2	Head3
A	B	C
D	E	F

```

<TABLE BORDER>
<TR>
    <TH>Head1</TH> <TH>Head2</TH> <TH>Head3</TH>
</TR>
<TR>
    <TD>A</TD> <TD>B</TD> <TD>C</TD>
</TR>
<TR>
    <TD>D</TD> <TD>E</TD> <TD>F</TD>
</TR>
</TABLE>

```

ДЕМОНСТРАЦИЯ COLSPAN И ЗАГОЛОВКОВ

Head1		Head2	
A	B	C	D
E	F	G	H

```

<TABLE BORDER>
<TR>
    <TH COLSPAN=2>Head1</TH>
    <TH COLSPAN=2>Head2</TH>
</TR>

```

```

<TR>
  <TD>A</TD> <TD>B</TD> <TD>C</TD> <TD>D</TD>
</TR>
<TR>
  <TD>E</TD> <TD>F</TD> <TD>G</TD> <TD>H</TD>
</TR>
</TABLE>

```

ДЕМОНСТРАЦИЯ МНОЖЕСТВЕННЫХ ЗАГОЛОВКОВ, COLSPAN

Head1		Head2	
Head 3	Head 4	Head 5	Head 6
A	B	C	D
E	F	G	H

```

<TABLE BORDER>
<TR>
  <TH COLSPAN=2>Head1</TH>
  <TH COLSPAN=2>Head2</TH>
</TR>
<TR>
  <TH>Head 3</TH> <TH>Head 4</TH>
  <TH>Head 5</TH> <TH>Head 6</TH>
</TR>
<TR>
  <TD>A</TD> <TD>B</TD> <TD>C</TD> <TD>D</TD>
</TR>
<TR>
  <TD>E</TD> <TD>F</TD> <TD>G</TD> <TD>H</TD>
</TR>
</TABLE>

```

ДЕМОНСТРАЦИЯ БОКОВЫХ ЗАГОЛОВКОВ

Head1	Item 1	Item 2	Item 3
Head2	Item 4	Item 5	Item 6
Head3	Item 7	Item 8	Item 9

```

<TABLE BORDER>
<TR><TH>Head1</TH>
  <TD>Item 1</TD> <TD>Item 2</TD> <TD>Item 3</TD></TR>
<TR><TH>Head2</TH>
  <TD>Item 4</TD> <TD>Item 5</TD> <TD>Item 6</TD></TR>
<TR><TH>Head3</TH>
  <TD>Item 7</TD> <TD>Item 8</TD> <TD>Item 9</TD></TR>
</TABLE>

```

ДЕМОНСТРАЦИЯ БОКОВЫХ ЗАГОЛОВКОВ, ROWSPAN

Head1	Item 1	Item 2	Item 3	Item 4
	Item 5	Item 6	Item 7	Item 8
Head2	Item 9	Item 10	Item 3	Item 11

```

<TABLE BORDER>
<TR><TH ROWSPAN=2>Head1</TH>
  <TD>Item 1</TD> <TD>Item 2</TD> <TD>Item 3</TD> <TD>Item
4</TD>
</TR>
<TR><TD>Item 5</TD> <TD>Item 6</TD> <TD>Item 7</TD> <TD>Item
8</TD>
</TR>
<TR><TH>Head2</TH>
  <TD>Item 9</TD> <TD>Item 10</TD> <TD>Item 3</TD> <TD>Item 11
</TD>
</TR>
</TABLE>

```

ПРИМЕР ТАБЛИЦЫ ИСПОЛЬЗУЮЩЕЙ ВСЕ ЭТИ ТЭГИ

		Average	
		Height	Weight
Gender	Males	1.9	0.003
	Females	1.7	0.002

```

<TABLE BORDER>
<TR>  <TD><TH ROWSPAN=2></TH>
      <TH COLSPAN=2>Average</TH></TD>
</TR>
<TR>  <TD><TH>Height</TH><TH>Weight</TH></TD>
</TR>
<TR>  <TH ROWSPAN=2>Gender</TH>
      <TH>Males</TH><TD>1.9</TD><TD>0.003</TD>
</TR>
<TR>  <TH>Females</TH><TD>1.7</TD><TD>0.002</TD>
</TR>
</TABLE>

```

ИСПОЛЬЗОВАНИЕ ROWSPAN/COLSPAN

A	1	2
	3	4
C	D	

```

<TABLE BORDER>
<TR>

```

```

        <TD ALIGN=center ROWSPAN=2 COLSPAN=2>A</TD>
        <TD>1</TD>
        <TD>2</TD>
    </TR>
    <TR>
        <TD>3</TD>
        <TD>4</TD>
    </TR>
    <TR>
        <TD ALIGN=center ROWSPAN=2 COLSPAN=2>C</TD>
        <TD ALIGN=center ROWSPAN=2 COLSPAN=2>D</TD>
    </TR>
    <TR>
    </TR>
</TABLE>

```

Выравнивание, отступы, рамки

ТАБЛИЦА БЕЗ РАМКИ

```

                                Item 1      Item 2      Item 3
                                Item 4      Item 5
    <TABLE border="0">
    <TR>    <TD>Item 1</TD> <TD ROWSPAN=2>Item 2</TD> <TD>Item 3
</TD>
    </TR>
    <TR>    <TD>Item 4</TD> <TD>Item 5</TD>
    </TR>
</TABLE>

```

ТАБЛИЦА С РАМКой ШИРИНОЙ 10

1	Item	2	Item
3	Item	4	Item

```

    <TABLE BORDER=10>
    <TR>    <TD>Item 1</TD> <TD> Item 2</TD>
    </TR>
    <TR>    <TD>Item 3</TD> <TD>Item 4</TD>
    </TR>
</TABLE>

```

CELLPADDING И CELLSPACING

A	B	C
D	E	F

```

<TABLE BORDER CELLPADDING=10 CELLSPACING=0>
<TR>
  <TD>A</TD> <TD>B</TD> <TD>C</TD>
</TR>
<TR>
  <TD>D</TD> <TD>E</TD> <TD>F</TD>
</TR>
</TABLE>

```

A	B	C
D	E	F

```

<TABLE BORDER CELLPADDING=0 CELLSPACING=10>
<TR>
  <TD>A</TD> <TD>B</TD> <TD>C</TD>
</TR>
<TR>
  <TD>D</TD> <TD>E</TD> <TD>F</TD>
</TR>
</TABLE>

```

A	B	C
D	E	F

```

<TABLE BORDER CELLPADDING=10 CELLSPACING=10>
<TR>
  <TD>A</TD> <TD>B</TD> <TD>C</TD>
</TR>
<TR>
  <TD>D</TD> <TD>E</TD> <TD>F</TD>
</TR>
</TABLE>

```

A	B	C
D	E	F

```

<TABLE BORDER=5 CELLPADDING=10 CELLSPACING=10>
<TR>
  <TD>A</TD> <TD>B</TD> <TD>C</TD>
</TR>
<TR>
  <TD>D</TD> <TD>E</TD> <TD>F</TD>
</TR>
</TABLE>

```

ВЫРАВНИВАНИЕ, ЗАГОЛОВКИ И ПОДТАБЛИЦЫ

ДЕМОНСТРАЦИЯ МНОЖЕСТВЕННЫХ СТРОК В ТАБЛИЦЕ СТРОК В ТАБЛИЦЕ

January	February	March
This is cell 1	Cell 2	Another cell, cell 3
Cell 4	and now this is cell 5	Cell 6

```
<TABLE BORDER>
<TR>
  <TH>January</TH>
  <TH>February</TH>
  <TH>March</TH>
</TR>
<TR>
  <TD>This is cell 1</TD>
  <TD>Cell 2</TD>
  <TD>Another cell,<br> cell 3</TD>
</TR>
<TR>
  <TD>Cell 4</TD>
  <TD>and now this<br>is cell 5</TD>
  <TD>Cell 6</TD>
</TR>
</TABLE>
```

ALIGN=LEFT|RIGHT|CENTER

может применяться к отдельным ячейкам или всей строке

January	February	March
all aligned center	Cell 2	Another cell, cell 3
aligned right	aligned to center	default, aligned left

```
<TABLE BORDER>
<TR>
  <TH>January</TH>
  <TH>February</TH>
  <TH>March</TH>
</TR>
<TR ALIGN=center>
  <TD>all aligned center</TD>
```

```

    <TD>Cell 2</TD>
    <TD>Another cell,<br> cell 3</TD>
</TR>
<TR>
    <TD ALIGN=right>aligned right</TD>
    <TD ALIGN=center>aligned to center</TD>
    <TD>default,<br>aligned left</TD>
</TR>
</TABLE>

```

VALIGN=TOP|BOTTOM|MIDDLE

может применяться к отдельным ячейкам или всей строке

January	February	March
all aligned to top	and now this is cell 2	Cell 3
aligned to the top	aligned to the bottom	default alignment, center

```

<TABLE BORDER>
<TR>
    <TH>January</TH>
    <TH>February</TH>
    <TH>March</TH>
</TR>
<TR VALIGN=top>
    <TD>all aligned to top</TD>
    <TD>and now this<br>is cell 2</TD>
    <TD>Cell 3</TD>
</TR>
<TR>
    <TD VALIGN=top>aligned to the top</TD>
    <TD VALIGN=bottom>aligned to the bottom</TD>
    <TD>default alignment,<br>center</TD>
</TR>
</TABLE>

```

CAPTION=TOP|BOTTOM

Верхний заголовок		
January	February	March
This is cell 1	Cell 2	Another cell, cell 3

```

<TABLE BORDER>
<CAPTION ALIGN=top>A top CAPTION</CAPTION>

```

```

<TR>
  <TH>January</TH>
  <TH>February</TH>
  <TH>March</TH>
</TR>
<TR>
  <TD>This is cell 1</TD>
  <TD>Cell 2</TD>
  <TD>Another cell,<br> cell 3</TD>
</TR>
</TABLE>

```

Нижний заголовок		
January	February	March
This is cell 1	Cell 2	Another cell, cell 3

```

<TABLE BORDER>
<CAPTION ALIGN=bottom>A bottom CAPTION</CAPTION>
<TR>
  <TH>January</TH>
  <TH>February</TH>
  <TH>March</TH>
</TR>
<TR>
  <TD>This is cell 1</TD>
  <TD>Cell 2</TD>
  <TD>Another cell,<br> cell 3</TD>
</TR>
</TABLE>

```

ВЛОЖЕННЫЕ ТАБЛИЦЫ: ТАБЛИЦА ABCD ВНУТРИ ТАБЛИЦЫ 123456

1	2	3	
		A	B
		C	D
4	5	6	

```

<TABLE BORDER>
<TR> <!-- ROW 1, TABLE 1 -->
  <TD>1</TD>
  <TD>2</TD>
  <TD>3
  <TABLE BORDER>
    <TR> <!-- ROW 1, TABLE 2 -->

```

```

        <TD>A</TD>
        <TD>B</TD>
    </TR>
    <TR> <!-- ROW 2, TABLE 2 -->
        <TD>C</TD>
        <TD>D</TD>
    </TR>
</TABLE>
</TD>
</TR>
<TR> <!-- ROW 2, TABLE 1 -->
    <TD>4</TD>
    <TD>5</TD>
    <TD>6</TD>
</TR>
</TABLE>

```

ЗАДАНИЕ ШИРИНЫ ТАБЛИЦЫ

ШИРИНА 50%

Width=50%	Width=50%
3	4

```

<TABLE BORDER WIDTH="50%">
<TR><TD>Width=50%</TD><TD>Width=50%</TD>
</TR>
<TR><TD>3</TD><TD>4</TD>
</TR>
</TABLE>

```

Item width affects cell size	2
3	4

```

<TABLE BORDER WIDTH="50%">
<TR><TD>Item width affects cell size</TD><TD>2</TD>
</TR>
<TR><TD>3</TD><TD>4</TD>
</TR>
</TABLE>

```

WIDTH=100%	This is item 2
3	4

```

<TABLE BORDER WIDTH="100%">
<TR><TD>WIDTH=100%</TD><TD>This is item 2</TD>
</TR>
<TR><TD>3</TD><TD>4</TD>

```

```
</TR>
</TABLE>
```

ЦЕНТРИРОВАНИЕ ТАБЛИЦЫ НА СТРАНИЦЕ

A	B	C
D	E	F

```
<CENTER>
<TABLE BORDER WIDTH="50%">
  <TR>
    <TD>A</TD> <TD>B</TD> <TD>C</TD>
  </TR>
  <TR>
    <TD>D</TD> <TD>E</TD> <TD>F</TD>
  </TR>
</TABLE>
</CENTER>
```

ШИРИНА ТАБЛИЦЫ И ВЛОЖЕННЫЕ ТАБЛИЦЫ

Item 1		Item 2
Item A	Item B	Item 4

```
<TABLE BORDER WIDTH="50%">
<TR><TD>Item 1</TD><TD>Item 2</TD>
</TR>
<TR><TD>
  <TABLE BORDER WIDTH=100%>
    <TR><TD>Item A</TD><TD>Item B</TD>
    </TR>
  </TABLE>
</TD>
<TD>Item 4</TD>
</TR>
</TABLE>
```

HEIGHT=15%

HEIGHT=15%	Item 2
3	4

```
<TABLE BORDER WIDTH="50%" HEIGHT="15%">
<TR><TD>HEIGHT=15%</TD> <TD>Item 2</TD>
</TR>
<TR><TD>3</TD><TD>4</TD>
</TR>
</TABLE>
```

Выполнить задание

Постройте следующие таблицы с помощью тегов html. Таблицы представлены во вложенном файле

Отчетность по практической работе

Необходимо написать отчет и подвесить в своем профиле в систему университета Ecampus.ncfu.ru (и/или отправить на адрес zamira.ncfu@gmail.com) детальное сообщение о проделанной работе, прикрепив страницу html с созданными таблицами.

В строке 'Тема' письма, указать номер практической работы и ваше ФИО.

Письмо должно содержать приветствие, что выполнено - по пунктам и что не выполнено, подпись автора.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания

по организации и проведению самостоятельной работы
по дисциплине

«Информационные технологии в профессиональной деятельности»

для студентов направления подготовки

09.03.02 «Информационные системы и технологии»

направленность (профиль)

**«Прикладное программирование в интеллектуальных
информационных системах»**

Ставрополь, 2026

Введение

1. Цель и задачи освоения дисциплины

Целью изучения дисциплины «Информационные технологии в профессиональной деятельности» является изучение таких вопросов и понятий как информация, содержание информационной технологии как составной части информатики, эволюция информационных технологий. Изучаются базовые информационные процессы, их характеристика, базовые Информационные технологии в профессиональной деятельности, прикладные Информационные технологии в профессиональной деятельности, стандарты пользовательского интерфейса и др.

Задачи дисциплины:

1. изучение понятий информационных технологий в профессиональной деятельности;
2. изучение основ инфокоммуникационных технологий;
3. изучение основ технологий разработки web-сайтов.

2. Место дисциплины в структуре ОП ВО бакалавриата

Дисциплина «Информационные технологии в профессиональной деятельности» относится к дисциплинам (модулям) Б1, часть, формируемая участниками образовательных отношений Б1.О.10 и изучается в 3 семестре.

3. Связь с предшествующими дисциплинами

Для успешного освоения дисциплины студенты используют знания, умения, навыки, способы деятельности, полученные и сформированные на предыдущих ступенях обучения и при изучении дисциплины «Информационные технологии в профессиональной деятельности».

4. Связь с последующими дисциплинами

Освоение данной дисциплины является необходимой основой для изучения дисциплин:

- Корпоративные информационные системы
- Интеллектуальные информационные системы
- Современные технологии программирования

В результате освоения дисциплины обучающийся должен:

Код, формулировка компетенции	Код, формулировка индикатора	Планируемые результаты обучения по дисциплине (модулю), характеризующие этапы формирования компетенций, индикаторов
-------------------------------------	---------------------------------	--

ОПК-2 Способен понимать принципы работы современных информационных технологий и программных средств, в том числе отечественного производства, и использовать их при решении задач профессиональной деятельности	ОПК-2_{ид-2.1} – Демонстрирует знания современных информационных технологий и программных средств, в том числе отечественного производства при решении задач профессиональной деятельности	обладает глубокими знаниями современных информационных технологий, способен применять передовые методы и программные средства для решения сложных профессиональных задач; знаком с программным обеспечением отечественного производства, умеет выбирать наиболее подходящие инструменты для решения конкретных проблем.
	ОПК-2_{ид-2.2} – Выбирает современные информационные технологии и программные средства, в том числе отечественного производства при решении задач профессиональной деятельности	способен выбирать и применять сложные информационные технологии и передовые программные средства в решении профессиональных задач, в том числе используя программное обеспечение отечественного производства для достижения наилучших результатов
	ОПК-2_{ид-2.3} – Применяет принципы работы современных информационных технологий и программных средств, в том числе отечественного производства	обладает глубокими навыками работы с современными информационными технологиями и программным обеспечением, включая программное обеспечение отечественного производства; может применять передовые методы и инструменты для решения сложных задач и способен интегрировать различные программные продукты в свою работу

	ОПК-2 ^{ид-2.4} – Использует современные информационные технологий и программные средства, в том числе отечественного производства, при решении задач профессиональной деятельности	обладает глубокими знаниями и широким спектром навыков в области информационных технологий и программного обеспечения. Способен применять передовые технологии и программные продукты для решения сложных и специфических задач в профессиональной деятельности, а также интегрировать различные программные средства отечественного и зарубежного производства в свою работу
--	---	---

5. Организационно-методические рекомендации по освоению дисциплины

Самостоятельная работа студентов является важнейшим условием формирования научного способа познания. Она проводится накануне каждого лабораторного занятия и включает изучение необходимого для выполнения лабораторной работы теоретического материала, а также подготовку отчета по выполненным на лабораторном занятии заданиям.

Подготовленный материал оформляется в виде отчета.

Самостоятельные занятия (СЗ) являются одной из активных форм обучения.

Самостоятельные занятия по дисциплине «Информационные технологии в профессиональной деятельности» имеют целью:

- закрепить и углубить знания, полученные студентами на лекциях и в процессе лабораторных занятий;
- привить практические навыки при разработке, тестировании Web-приложений.

Самостоятельные занятия проводятся в специализированных аудиториях, оборудованных СBT и возможностью пользования Интернет-ресурсами, в библиотеке, а также дома.

Предлагаемые методические рекомендации содержат информацию для студентов, необходимую при подготовке и проведении лабораторных занятий по дисциплине «Информационные технологии в профессиональной деятельности».

6. Методические указания по выполнению самостоятельной работы студентов

Методические указания включают следующие разделы:

- цель работы;
- задание, которое должно быть выполнено студентом в результате проведения самостоятельной работы;
- варианты индивидуальных заданий;
- основные теоретические положения, необходимые для выполнения задания, они должны быть краткими и содержать ссылки на литературу, в которой эти положения изложены в объеме, достаточном для выполнения самостоятельной работы;
- этапы выполнения задания с указанием конкретных сроков выполнения каждого из этапов и всего задания в целом;
- требования к оформлению графической и текстовой части самостоятельной работы;
- пример выполнения одного из вариантов задания и оформления отчета;
- библиографический список использованных источников

7. Содержание самостоятельной работы

Наименование разделов и тем дисциплины, их краткое содержание; вид самостоятельной работы представлено в таблице 2.

Таблица 2 – Наименование разделов и тем дисциплины, их краткое содержание; вид самостоятельной работы

№	Наименование разделов и тем дисциплины, их краткое содержание; вид самостоятельной работы	Форма контроля
	3 семестр	
1	подготовка к лабораторным занятиям	защита
2	Самостоятельное изучение вопросов: Беспроводные технологии организации каналов связи Основные меры и способы защиты информации в информационных технологиях	опрос

3	Самостоятельная разработка статических страниц веб-сайта	Защита проекта на зачете
---	--	--------------------------

Наименование тем дисциплины, цель, форма контроля, задания, требования к оформлению, перечень литературных источников представлен в таблице 3.

Таблица 3 – Наименование тем дисциплины, цель, форма контроля, задания и требования к оформлению самостоятельной работы

Название темы	Цель	Форма контроля	Задания для СРС
Подготовка к лекциям и лабораторным занятиям	изучить основные современные информационных технологий сбора, хранения, обработки и передачи данных	индивидуальное собеседование	Сформулировать ответы на контрольные вопросы
Разработка собственного статического веб-сайта	Получить навыки разработки статического веб-сайта		сформулировать и ответить на вопросы для контроля владения компетенциями данного раздела программы

8. Примерная тематика заданий для самостоятельной работы студентов

Примерные варианты веб-сайта (веб-приложения) по дисциплине «Информационные технологии в профессиональной деятельности»:

1. Разработка web-приложения «Расписание»
2. Разработка web-приложения «Университет»
3. Разработка web-приложения «Хлебозавод»
4. Разработка web-приложения «Кондитерская»
5. Разработка web-приложения «Компания по производству окон»
6. Разработка web-приложения «Школа»
7. Разработка web-приложения «Ювелирный магазин»
8. Разработка web-приложения «Обувной бутик»

9. Разработка web-приложения «Магазин одежды»
10. Разработка web-приложения «Туристическое агентство»
11. Разработка web-приложения «Аптека»
12. Разработка web-приложения «Автосервис»
13. Разработка web-приложения «Компьютерный салон»
14. Разработка web-приложения «Художественная школа»
15. Разработка web-приложения «Театр»
16. Разработка web-приложения «Спортивные товары»
17. Разработка web-приложения «Кинотеатр»
18. Разработка web-приложения «Автосалон»
19. Разработка web-приложения «Туризм»
20. Разработка web-приложения «Гостиница»
21. Разработка web-приложения «Электронная библиотека»
22. Разработка web-приложения «АЗС»
23. Разработка web-приложения «Ресторан»
24. Разработка web-приложения «Мобильный оператор»
25. Разработка web-приложения «Книжный магазин»
26. Разработка web-приложения «Школа иностранного языка»
27. Разработка web-приложения «Юридическое агентство»
28. Разработка web-приложения «Веломагазин»
29. Разработка web-приложения «Продажа фруктов»
30. Разработка web-приложения «Языки программирования»
31. Разработка web-приложения «Магазин видеоигр»
32. Разработка web-приложения «Компьютерная техника»
33. Разработка web-приложения «Поликлиника»
34. Разработка web-приложения «Частная радио (теле) компания».
35. Разработка web-приложения «Жилищно-коммунальное хозяйство»
36. Разработка web-приложения «Ремонтные работы»
37. Разработка web-приложения «Платная стоматологическая клиника»
38. Разработка web-приложения «Ветеринарная поликлиника»
39. Разработка web-приложения «Сувениры»
40. Разработка web-приложения «Канцтовары»

41. Разработка web-приложения «Лаборатория ландшафтного дизайна»
42. Разработка web-приложения «Лаборатория квартирному дизайну»
43. Разработка web-приложения «Компьютерные комплектующие»
44. Разработка web-приложения «Электронный секретарь»
45. Разработка web-приложения «Банкротство»
46. Разработка web-приложения «Аукцион»
47. Разработка web-приложения «Принтеры»
48. Разработка web-приложения «Городское озеленение»
49. Разработка web-приложения «Памятники архитектуры»

9. Организация контроля знаний студентов

9.1 Формы контроля знаний студентов

Контроль и оценка знаний, умений и навыков студентов осуществляется на лабораторных занятиях, консультациях, получении зачета с оценкой. В ходе контроля знаний преподаватель оценивает понимание студентом содержания дисциплины «Информационные технологии в профессиональной деятельности», его способность разрабатывать и тестировать веб-приложение, применять веб-технологии в разработке.

Контроль знаний студентов может осуществляться в следующих формах:

- текущий контроль знаний;
- итоговый контроль знаний.

Текущий контроль знаний студентов имеет целью: дать оценку работы каждого студента по усвоению им учебного материала, выявить недостатки в его подготовке и оказать практическую помощь в их устранении;

Основными формами текущего контроля знаний студентов являются:

- устный контрольный опрос;
- защита лабораторной работы.

Устный контрольный опрос студентов проводится на лекциях (и лабораторных занятиях). По его результатам преподаватель оценивает качество подготовки студента к занятию.

На лабораторных занятиях знания и практические навыки студентов оцениваются по 5-балльной системе.

9.2. Рекомендации по подготовке к зачету

Подготовка к зачету начинается с начала изучения дисциплины.

Необходимо посещать все лекции и лабораторные занятия.

Зачет проводится в 4 семестре после защиты всех лабораторных работ в объеме учебной программы и выполненной самостоятельной работы.

10 Перечень рекомендуемой литературы

а. Основная литература

1. Основы информационных технологий [Электронный ресурс]: учебное пособие/ Г.И. Киреева [и др.].– Электрон. текстовые данные.– Саратов: Профобразование, 2017.– 272 с.– Режим доступа: <http://www.iprbookshop.ru/63942.html>.– ЭБС «IPRbooks».

2. Цветкова, А. В. Информатика и Информационные технологии в профессиональной деятельности Электронный ресурс : Учебное пособие для СПО / А. В. Цветкова. - Информатика и Информационные технологии в профессиональной деятельности, 2020-08-30. - Саратов : Научная книга, 2019. - 190 с. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 978-5-9758-1891-1.

3. Сузи, Р.А. Язык программирования Python Электронный ресурс : учебное пособие / Р.А. Сузи. - Язык программирования Python, 2020-07-28. - Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. - 350 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 5-9556-0058-2

Дополнительная литература:

1. Куприянов Д. В. Информационное и технологическое обеспечение профессиональной деятельности : учебник и практикум / Д. В. Куприянов ; Финансовый ун-т при Правительстве Рос. Федерации. - Москва : Юрайт, 2017. - 255 с.

2. Кислицын Е. В. Инструменты обработки и анализа корпоративных данных : учеб. пособие / Е. В. Кислицын, Е. М. Кочкина, Е. В. Радковская. - Екатеринбург : Изд-во УрГЭУ, 2021. - 227 с.

3. Информатика и информационные технологии : учеб. пособие / С. В. Бегичева, Е. В. Долженкова, И. Е. Жуковская [и др.] ; под общ. ред. Д. М. Назарова. - Екатеринбург : Изд-во УрГЭУ, 2020. - 219 с.

4. Информационные технологии в профессиональной деятельности : учебное пособие / А. Н. Гвоздева, И. Ю. Лаврентьева. – М. : Форум, 2018. – 320 с.

5. Информационные технологии в управлении : учебник для вузов / В. В. Трофимов. – М. : Юрайт, 2019. – 478 с.

10.3. Методическая литература

УМК по дисциплине «Информационные технологии в

профессиональной деятельности» (электронных вариант).

10.4. Интернет-ресурсы

1	https://biblioclub.ru/
2	https://www.elibrary.ru/
3	https://metanit.com/

11. Материально-техническое обеспечение дисциплины

- индивидуальное взаимодействие со студентами по электронной почте для предварительного ознакомления с их разработками при подготовке к аудиторным занятиям;
- использование на лекциях и лабораторных занятиях мультимедийного оборудования для демонстрации листингов программного кода, скриншотов выполненных запросов, работы программ и пр.;
- включение в лабораторные работы индивидуального поиска, систематизации и анализа информации через Интернет;
- авторские презентации к лекциям;
- лекционная аудитория должна быть оборудована мультимедиа проектором;
- Лабораторные работы рассчитаны на компьютеры, имеющие доступ к Internet.

12.1. Программное обеспечение:

1	Notepad+
2	Windows Movie Maker

12.2. Материально-техническое обеспечение дисциплины (модуля)

Мультимедиа аудитория: проектор, акустическая система, аудио-усилитель, ноутбук, Internet.

Компьютерный класс. Высокопроизводительная ЭВМ, локальная вычислительная сеть.