Попытка написать некое пособие по лабам по HTML. Пока совершенно неясна структура материала.

Л\р №1.

Организация рабочего места разработчика и необходимые общие сведения для начала работы.

1. Изучить состав ПО на вашем компьютере, определить, какие текстовые редакторы и интернет-браузеры у вас установлены.
   В помощь — возможный список для OS Linux:
   1) редакторы: Geany (рекомендация автора), BlueFish, Quanta+, Kwrite, Kate;
   2) браузеры: Chromium, Google Chrome, FireFox, Opera, Konqueror.
   Возможный список для OS Windows:
   1) редакторы: Geany (рекомендация автора), NotePad;
   2) браузеры: Google Chrome, FireFox, Opera, Internet Explorer.
   Более полные списки программ можно найти, например, в Википедии.
   Замечание: офисные программы Microsoft Word, OpenOffice Writer относятся к классу *текстовых процессоров* и для выполнения л\р по данной теме непригодны.
2. Создать в вашей персональной папке отдельную папку Site для хранения всех файлов, относящихся к выполнению л\р по данной теме. В этой папке создать пустой текстовый файл с именем 1.html и открыть его (с помощью ПКМ) в выбранном вами текстовом редакторе. Пользователи OS Windows должны установить в редакторе кодировку utf-8. Вписать произвольную фразу по русски, сохранить файл (но не закрывать!), открыть его же (с помощью ПКМ) в выбранном вами интернет-браузере. Убедиться, что без перевода браузера в режим чтения кодировки utf-8 русский текст прочитать невозможно. Вернуться в редактор и вписать *перед текстом* инструкцию <meta charset=utf-8>, сохранить файл, переключиться в окно браузера, обновить окно браузера, убедиться, что русский текст можно прочитать.
   Замечание: необходимость использования кодировки utf-8 описана в специальной литературе, на занятиях подробно не обсуждается.
3. Изучить инструкции по рациональным приемам работы с компьютером. Здесь — коротко:
   при выполнении л\р по данной теме происходит интенсивная работа с редактором, файловой системой, оконным менеджером, браузером; основные операции: выделить фрагмент, копировать или вырезать фрагмент в буфер обмена, вставить фрагмент из буфера обмена, сохранить файл, переключиться в окно браузера, обновить окно браузера — эти операции зачастую выполняются до нескольких раз в одну минуту в течение всего занятия, выполнение этих операций с помощью главных меню используемых вами программ или контекстных меню, вызываемых с помощью ПКМ, сильно замедляет работу, это может оказаться критическим для выполнения работ по данной теме. Владение горячими клавишами (hot keys) позволит вам быстро вносить необходимые изменения и успевать за объяснениями преподавателя.
4. Изучить ресурсы Интернет по языкам HTML и CSS. Возможные запросы для начала работы: «учебник html», «учебник css». Обратить внимание на время обновления материала, удобство работы с материалом на выбранных сайтах, наличие примеров, доступность изложения.

Л\р №2.

Основные методы создания структуры материала.

1. Веб-сайт представляет собой набор отдельных страниц, каждая хранится в отдельном файле (пример: 1.html) и разнообразных ресурсов: картинок (картинки не являются частью страницы и хранятся каждая в отдельном файле, страницы только содержат ссылки на картинки), видео- и аудиофайлов, стилевых файлов, программ (скриптов) и т.д. В свою очередь, страница представляет собой некоторое содержание — *контент* (блоки текста, заголовки, списки, таблицы и т.д.), структурированное с помощью раскладки в *контейнеры*. Контейнеры могут неограниченно вкладываться друг в друга, образуя иерархическую систему, построение иерархии должно соответствовать логике излагаемого материала. Рассмотрим организацию типичного контейнера — заголовка 1 уровня:
   <h1>Глава 1. Начало всех начал</h1>
   Здесь:
   <h1> - тэг, открывающий контейнер,
   </h1> - тэг, закрывающий контейнер,
   слова « Глава 1. Начало всех начал» - содержимое контейнера (контент).
   Впишите в свой файл 1.html заголовок с произвольным (вашим, авторским) содержанием, просмотрите результат в браузере. Согласно логике материала, заголовок следует поместить перед текстом, который вы набрали в прошлый раз, но после meta-тэга.
2. Для грамотной организации кода, каждая страница должна иметь следующую структуру обязательных контейнеров:

```
<!DOCTYPE html>
<html>
<head>
        <title> … … … </title>
        <meta charset=utf-8>
        <link rel=stylesheet href=1.css>
</head>
<body>




</body>
</html>
```

   Коротко: doctype устанавливает версию языка html, а именно, версию html5, нужен для корректного отображения стилевых свойств, head - «голова» сайта (не путать с «шапкой» сайта!), содержит служебную информацию, в ней: title - заголовок, который будет отображаться на ярлыке вкладки браузера, link rel=stylesheet - подключение стилевого файла, причем имя этого файла - 1.css дано только в качестве примера, вы можете использовать любое другое имя (подробнее в следующей главе), body - «тело» сайта, содержит контент, то есть все, что будет отображено в окне браузера, то, что увидит пользователь. Большое пустое место оставлено для наглядности — именно сюда следует перенести то содержание, которое у вас уже есть. За пределами контейнера body писать ничего не следует!

Вы можете скопировать эту структуру отсюда прямо в ваш файл 1.html и отредактировать должным образом.

3. Переносы строк. «Урок поэзии».
   Напишите (не забудьте, писать надо только в промежутке между тэгами <body> и </body>) любое 4-стишие, просмотрите в браузере. Убедитесь, что браузер выстраивает весь текст в одну строку, не обращая внимания на расположение строк в файле 1.html. Это правильное поведение! Для того, чтобы организовать перенос строки в браузере, в тексте нужно вставить тэг
 (англ. break). Расставьте «брейки» в вашем тексте, чтобы получилось правильное 4-стишие. Обратите внимание, что заголовок отделялся от текста самостоятельно, без использования «брейков».
   Самостоятельно отделяются от текста: заголовки, списки, таблицы, абзацы, блоки (о них — немного позже); не отделяются: картинки и гиперссылки.
   Замечание: использование «брейка», кроме как для организации стихов, нежелательно. Более грамотно все фрагменты текста заключать в контейнеры «абзац»: … … … .
4. Вставка картинок.
   1) найдите в Интернет любую картинку небольших размеров (изменение размера картинки требует дополнительных усилий и вообще нежелательно), сохраните ее и поместите в вашу папку Site;
   2) переименуйте картинку, дав ей максимально простое имя, например, 1.jpg;
   3) впишите в ваш код (до или после стиха — как впишете, так и будет расположено) следующую инструкцию:

   <img src=1.jpg>

   (здесь 1.jpg - это только пример имени картинки, если вы назвали ее иначе, то впишите то имя, которое дали вы), «брейки» нужны для отделения картинки от текста, если вы не использовали абзацы;
   4) просмотрите в браузере, убедитесь, что картинка отобразилась.
5. Вставка гиперссылок.
   Пример: гиперссылка на сайт ТГПУ:

   Сайт ТГПУ
   Здесь http://tspu.edu.ru — адрес, он не виден пользователю,
   слова «Сайт ТГПУ» - текст гиперссылки, то, что пользователь увидит на экране.
   Замечание: адрес должен быть указан полностью и без ошибок, поэтому нежелательно набирать его вручную на клавиатуре, его копируют из адресной строки браузера и вставляют в код.
   Вставьте в свою страницу ссылку на любой избранный вами сайт. Не забудьте, что гиперссылка не отделяется от текста самостоятельно.
6. Вставка таблиц (для особо продвинутых, малоактуально).
   <table>
   <tr><td> … </td> … … </tr>
   <tr><td> … </td> … … </tr>
   … …
   </table>
   Здесь:
   <tr> - строка таблицы
   <td> - ячейка в строке. Число ячеек в строках должно быть одинаковым, иначе таблицу «перекосит», следить за этим — самостоятельно. Ячейкам, при желании, можно придавать размеры, выравнивание текста, фон и т. д.
   Типичное применение: прайс-листы, таблицы каких-либо данных.

Л\р №3.

Основы верстки.

1. Верстка веб-страницы опирается на ту структуру, которая создана в html-файле, но осуществляется другим языком — специальным языком стилей, CSS (cascading style sheets). Для привязки css-файла к веб странице служит инструкция, которую мы уже написали ранее:
   <link rel=stylesheet href=1.css>
   Таким образом, стилевые свойства элементов веб-страницы записываются в отдельном стилевом файле, 1.css (его можно назвать любым именем с расширением .css, и именно это имя должно быть вписано в инструкцию «link»). К одной странице можно прикрепить более одного стилевого файла; один стилевой файл может управлять любым числом страниц.
   Замечание: язык html имеет собственные средства для верстки, но пользоваться ими крайне нежелательно (здесь это не обсуждается).
2. Пример стилевого свойства.
   Создайте в своей папке Site файл 1.css, откройте его выбранным вами редактором и впишите в него следующую инструкцию:
   body {background: lightgrey; color: green}
   Сохраните этот файл и просмотрите страницу в браузере. Фон страницы должен окраситься в светло-серый цвет, текст на странице (включая заголовки и списки, но не гиперссылки) — в зеленый. Данные цвета приведены для примера и не представляют самостоятельной ценности, желательно сразу заменить их на ваши авторские.
   Пример показывает, как пишутся стилевые свойства (*синтаксис*): сначала пишется имя элемента, к которому свойство применяется, сами свойства записываются в фигурных скобках через точку с запятой, значения присваиваются с помощью двоеточия.
3. Прямоугольный блок .
   Основой верстки веб-страниц можно смело назвать элемент . Не будучи стилизован, он представляет собой невидимый прямоугольник, имеющий блочное поведение (то есть отделяющийся по вертикали от простого текста). Блок образует контейнер, который может содержать любой контент, отображение контента определяется теми стилевыми свойствами, которые связаны с конкретным блоком . Одна страница может содержать множество блоков самого разного назначения и стиля, поэтому создаются классы однотипных блоков, и стилевые свойства приписываются классам, а не блокам. В html-файле классы приписываются блокам следующим образом — при помощи атрибута class у открывающего тэга, например:
    … … …
   Здесь left_sidebar - имя класса, оно авторское, очевидно, имеется в виду левая боковая колонка. Желательно, чтобы имена классов отражали суть, а не были пустыми сочетаниями букв, в противном случае вы сами быстро запутаетесь в своем коде. В css-файле стилевые свойства придаются классу так же, как и любому элементу, имя класса записывается с начальной точкой, например:
   .left_sidebar {color: blue}
   Самому элементу придавать какие-либо свойства нежелательно, он должен сохранять универсальность.
   Вложите весь контент вашей страницы в блок … … , так чтобы в body кроме этого блока больше ничего не было. Напомню, что за пределами body вообще не следует что-либо писать. В стилевом файле впишите:
   .content{

```
position: absolute;
top: 300px;
left: 400px;
width: 700px;
border: 1px solid;
border-radius: 30px;
background: white;
padding: 20px;
font: 15px Sans;
}
```

Сохраните css-файл и просмотрите страницу. Измените все значения на ваши авторские.

Замечание: в данном примере показан «хороший стиль» написания стилевых файлов: каждое свойство — на отдельной строчке, имя элемента (или класса) — на отдельной строчке, свойства сдвинуты вправо (проще всего клавишей Tab). Нарушение этих правил помешает вам ориентироваться в вашем же коде и получать помощь от посторонних.

Очень краткий справочник стилевых свойств (с примерами значений)

- background: lightgrey - цвет фона (светлосерый)
- background: url(1.jpg) - фоновый рисунок (файл 1.jpg)
- background: linear-gradient(to top, grey, white) - градиентный фон (снизу вверх от серого к белому)
- color: blue - цвет текста (синий)
- padding: 10px - внутренний отступ, от текста до рамки (10 пикселей)
- margin: 10px - внешний отступ от рамки до внешнего окружения блока (10 пикселей)
- border: 1px solid blue - рамка блока (1 пиксель, сплошная, синяя)
- text-align: center - выравнивание текста в блоке (по центру)
- font: 15px bold Arial - шрифт (15 пикселей, жирный, гарнитура Ариал)
- border-radius: 5px - закругление уголков блока (радиус 5 пикселей)
- position: absolute; top: 100px; left: 200px - абсолютно позиционированный блок (100 пикс. от верхнего края и 200 - от левого)
- position: relative - относительное позиционирование
- float: left - блок, прижатый к левому краю содержащего блока, обтекается текстом

Л\р №4.

Создание модульной сетки сайта.

1. Недостатки абсолютного позиционирования.
   В л\р №3 мы рассмотрели создание абсолютно позиционированного блока . Абсолютное позиционирование — простой и ясный, но редкий прием из-за того, что такие блоки «не видят» своего окружения, в случае неверного расчета они наложатся на другие блоки, что недопустимо. Обратно, и окружающие блоки игнорируют наличие абсолютно позиционированного блока, как если бы его здесь не было. В условиях, когда мы не можем указать точные размеры блоков (например, если они содержат переменный контент), такое позиционирование неприемлемо. Оно хорошо подходит, например, для создания выносных элементов.
2. Статические блоки.
   Статическое позиционирование является поведением блоков по умолчанию.

Статический блок по ширине занимает все доступное пространство, а его высота определяется содержимым. Статические блоки выкладываются на страницу друг под другом в том порядке, в каком они записаны в html-файле (так называемый *прямой поток*). Мы это уже наблюдали на примере заголовка <h1> - как говорилось, он имеет блочное поведение.
Замечание: попытка вложить внутрь статического блока абсолютно позиционированный блок не приведет к успеху — компьютер не сможет вычислить его координаты, так как статический блок раскладывается по принципу «как получилось» и определенных координат не имеет.

3. Относительное позиционирование.
   Блоки со свойством position: relative ведут себя совершенно так же, как статические, но позволяют позиционировать вложенные блоки относительно себя. О других свойствах относительного позиционирования можно почитать в специальной литературе.
4. «Флоаты».
   Свойство блока float: left (или float: right) применяется для создания небольших блоков, занимающих не всю ширину текста, прижатых к краю так, чтобы текст обтекал эти блоки. Изначально «флоат» был задуман для вставки иллюстраций в большие тексты. Однако, «флоат» часто применяется для создания колонок текста.
5. Определение модульной сетки сайта.
   Практически любая страница, кроме уникальных дизайнерских разработок с целью удивить зрителя, имеет следующие блоки:
   1) «шапку» (header, не путать с <head>!), содержащую название, логотип, часто - горизонтальное меню и поисковую форму;
   2) блок основного содержания (content или wrapper — это устоявшиеся среди разработчиков названия);
   3) 1-2 боковых колонки с навигацией - «сайдбары» (sidebar);
   4) «подвал» (footer), содержащий копирайт, адрес разработчика и прочую второстепенную информацию.

Задание: найти в сети Интернет 3 внешне привлекательных сайта, поместить ссылки на них на своей странице (1.html)и кратко (в 10-15 предложениях) описать основные блоки на этих сайтах.

Л\р №5.

Создание центральной полосы с горизонтальным навигационным меню.

1. Создание центральной полосы.
   Существуют веские причины, по которым большинство сайтов не занимают всю ширину экрана, а представляют собой довольно узкую полосу в центре. Создать такую полосу волшебно легко:
   1) удалите записанный с чисто демонстрационными целями класс .content в вашем файле 1.css вместе со всеми его свойствами;
   2) впишите в этот файл следующее:

```
body {
    margin: 15px auto;
    width: 700px;
    }
```

   Сохраните файл 1.css и просмотрите страницу в браузере.
2. Сделаем центральную полосу более заметной.
   Сказанное в этом пункте полезно для придания нашей странице эстетической привлекательности. Впишите (можно скопировать прямо отсюда) в ваш стилевой файл следующее:
   1) html {background: lightgrey;}
   2) к свойствам тэга body добавьте:

```
    min-height: 100%;
    padding: 15px;
    border: 1px solid;
    border-radius: 30px;
    background: white;
    font: 15px Sans;
```

3. Создание навигационного меню.
   Внимание! Меню является частью структуры и вписывается в файл 1.html!
   Навигационное меню содержит ссылки на страницы самого сайта, а не на внешние ресурсы (Интернет), поэтому, прежде, чем его создать, нужно определить, какие страницы будет содержать ваш сайт. Предположим (это авторский выбор!), что мы создаем учебное пособие, возможные названия страниц:
   1) «Главная» - общее описание проекта, новости сайта, некоторая первичная, вводная информация;
   2) «Учебники» - статьи и ссылки на внешние учебные ресурсы;
   3) «Примеры»;
   4) «Задания»;
   5) «Об авторе».
   Напомним, что каждая страница сайта представляет отдельный файл. Главную мы создали как 1.html, определим имена остальных страниц как 2.html, 3.html, 4.html, 5.html. Тогда код навигационного меню можно создать в следующем виде:

```
<menu>
    <ul>
        <li><a href=1.html>Главная</a></li>
        <li><a href=2.html>Учебники</a></li>
        <li><a href=3.html>Примеры</a></li>
        <li><a href=4.html>Задания</a></li>
        <li><a href=5.html>Об авторе</a></li>
    </ul>
</menu>
```

В какое место body поместить этот код — вопрос дизайна, пока поместим его на самый верх body, до блока .
Замечание: элемент menu - современный элемент, появившийся в версии языка html5. Это блок, ничем, с точки зрения компьютера, не отличающийся от блока , разница между ними чисто *семантическая*.

4. Горизонтальное меню на кнопках.
   С точки зрения структуры, меню — список гиперссылок, внешнее оформление может быть самым разным и определяется стилями элементов menu, ul, li, a. Для начала оформим меню в виде горизонтального ряда кнопок со скругленными уголками, красивой градиентной заливкой, белым текстом увеличенного размера без подчеркивания. В стилевой файл внесите следующее:

```
menu {text-align: center;}
menu ul {
    padding: 0;
    margin: 0;}
menu li {
    display: inline-block;
    padding: 2 px 15px;
    margin: 0 10px;
    border: 1px solid;
    border-radius: 4px;
    background: linear-gradient(to top, grey, gold);
    }
menu a {
        text-decoration: none;
        color: white;
        font: 20px Sans;
        }
```

   Просмотрите, потом поменяйте, как минимум, цвета, на ваши авторские.
   Замечание: конструкция, употребленная в последнем примере: тэг1 тэг2 { … }, называется *контекстным селектором*. Она позволяет установить свойства контейнера «тэг2» только для того случая, когда он вложен в контейнер «тэг1», за пределами *родительского* «тэг2» останется нестилизованным. Действительно, нам незачем раскрашивать все гиперссылки на сайте в белый цвет! На белом фоне мы их просто не увидим. Мы это сделали только для гиперссылок навигационного меню. Такой подход создает некоторую альтернативу атрибуту class, рассмотренному выше.

Л\р №6.

Создание многостраничного сайта.

1. Создайте свой собственный стиль оформления страницы 1.html, в частности, поменяйте серый фон на полях страницы на фоновый рисунок (найдите его в Интернет, инструкцию по установке рисунка на фон элемента смотрите в Справочнике к Л\р №3).
2. Создайте копии файла 1.html, придав им имена 2.html, 3.html, 4.html, 5.html (эти имена прописаны в навигационном меню, вы можете изменить их на ваши авторские, однако, на отображение страницы это никак не повлияет — пользователь не видит имен файлов без специальных действий).
3. В каждом из вновь созданных файлов удалите содержимое блока и заполните своим авторским контентом.