Министерство образования Белгородской области
Областное государственное автономное
профессиональное образовательное учреждение
«Белгородский индустриальный колледж»

Рассмотрено
предметно-цикловой комиссией
Протокол заседания № ____
От «____»_____ 2022
Председатель цикловой комиссии
_____/ Третьяк И.Ю.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

МДК 08.01 «Проектирование и разработка интерфейсов пользователя»

ПМ.08 «Разработка дизайна веб-приложений»

по специальности

09.02.07 Информационные системы и программирование

Разработчик:

Солдатенко М.Н.
преподаватель ОГАПОУ
«Белгородский индустриальный
колледж»

Белгород 2022

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

МДК 08.01 Проектирование и разработка интерфейсов пользователя является специальным, формирующим базовые умения для получения выпускником профессиональных умений.

Методические рекомендации по выполнению лабораторных работ для студентов специальности 09.02.07 «Информационные системы и программирование» разработаны в соответствии с рабочей программой профессионального модуля **Разработка дизайна веб-приложений**, соответствуют требованиям Федерального государственного образовательного стандарта по специальностям среднего профессионального образования.

Целью методических рекомендаций по выполнению лабораторных работ является организация и управление работой студентов на практических занятиях при изучении данной дисциплины.

Методические рекомендации по выполнению лабораторных работ содержат тематический план и общие положения и требования к оформлению отчетов. Методические указания к каждой лабораторной работе включают в себя следующие элементы: название темы, цель занятия, ход работы, теоретическую часть, практическую часть (указания по выполнению) и контрольные вопросы.

Методические рекомендации содержат лабораторные работы, которые обеспечивают формирование базовых умений и навыков разработки интерфейса пользователя для веб-приложений с использованием современных стандартов; разработки дизайна веб-приложений в соответствии с требованиями заказчика, знаний норм и правил выбора стилистических решений; современных методик разработки графического интерфейса.

В лабораторных работах, приведенных в данных методических рекомендациях, содержатся как задания с подробными указаниями к выполнению, так и задания без алгоритма работы.

Методические рекомендации предназначены для студентов очной формы обучения специальности 09.02.07 «Информационные системы и программирование». По учебному плану по МДК 08.01 Проектирование и разработка интерфейсов пользователя на лабораторные работы студентов отводится 46 часов.

Методические рекомендации направлены на повышение мотивации учащихся к изучению междисциплинарного курса Проектирование и разработка интерфейсов пользователя, развитие гибкого логического и пространственного мышления учащихся, развитие профессиональных компетенций учащейся молодежи.

Тематический план

№	Название лабораторной работы	Количество часов
1,2,3	Применение тегов HTML при создании web-страниц	6
4,5	Создание формы на html-странице.	4
6,7,8	Форматирование web-страниц с использованием каскадных таблиц стилей	6
9,10,11	Динамические эффекты с использованием CSS	6
12,13,14	Вёрстка страниц web-сайта	6
15,16,17	Использование языка сценариев JavaScript при создании web-сайта	6
18	Подготовка и оптимизация графики на веб-странице. Создание баннера	2
19	Словарь схемы сайта. Логическая схема сайта	2
20	Разработка эскизов веб-приложения	2
21	Разработка прототипа дизайна веб-приложения	2
22	Разработка схемы интерфейса веб-приложения	2
23	Пользовательский интерфейс средствами CSS	2
	Всего	**46**

Лабораторная работа №№ 1-2-3
Создание веб-страниц средствами языка HTML

Цель: *Получить практические навыки при создании основы html-страницы; познакомиться с различными способами форматирования в html-документе; научиться создавать различные виды гиперссылок, внедрять изображения, аудио и видео файлы; выработать навык художественно-эстетического вкуса при оформлении Web-документа*

Ход работы:

1. Изучить теоретический материал.
2. Проработать примеры.
3. Выполнить задания в соответствии с указаниями.
4. Предъявить преподавателю результаты работы: проект и исходный код
5. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
6. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Для подготовки Web-страницы используется язык разметки гипертекста HTML (HyperText Markup Language). Как это следует из названия, HTML обладает средствами представления текста и изображений на экране в виде Web-страницы. HTML также позволяет работать с гипертекстовыми ссылками, с помощью которых осуществляется доступ к другим Web-страницам.

Стандарт HTML создан и поддерживается некоммерческой международной организацией W3C (World Wide Web Consortium).

Поскольку HTML-документ записывается в ASCII-формате, для создания Web-страницы может быть использован любой текстовый редактор, работающий с простым текстом, т.е. таким текстом, который не содержит служебной информации. Такими редакторами являются, например, NotePad (Блокнот).

После того, как с помощью текстового редактора HTML-документ создан, он должен быть сохранен на диске в виде файла с расширением .htm или .html. Это позволит после щелчка мыши по этому файлу запустить браузер, который осуществит формирование Web-страницы в соответствии с данным HTML-документом. Можно поступить по-другому. Сначала запустить браузер, а затем в его поле адреса (Address) указать путь к файлу с данным HTML-документом.

Браузером (browser, от англ. to browse - выслеживать) называется программа, которая выполняет следующие основные функции:

• после задания пользователем адреса Web-страницы и нажатия клавиши "переход" ("Go") устанавливает с помощью http-протокола соединение с Web-сервером, передает ему запрос и получает от него требуемую Web-страницу в виде HTML-документа;

• преобразует полученный HTML-документ в изображение не экране пользователя, которое и называется Web-страницей (если HTML-документ содержится на компьютере пользователя, то соединение с Web-сервером не является обязательным).

Примерами браузеров могут служить такие программы, как Microsoft Internet Explorer (MIE), Mozilla Firebox, Apple Safari, Netscape Navigator, Opera и др.

С помощью HTML можно создавать только статические Web-страницы. Для создания динамических страниц необходимо совместное использование HTML с такими языками, как JavaScript, PHP и др.

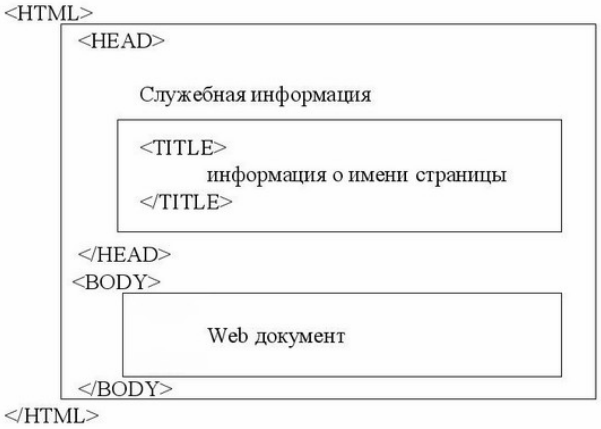
Структура Web-страницы

Web-страница или HTML-документ состоит из слов двух типов: служебных слов и собственно текста, выводимого на экран. Служебные слова называются тегами (tags) и берутся в угловые скобки < >. В литературе теги называют также дескрипторами и

операторами. Теги бывают двойными, называемые контейнерами (container), и одиночными. Примером контейнера служат теги, задающие начало HTML-документа: <HTML> и его окончание: </HTML>. Тег окончания контейнера отличается от тега его начала наличием символа "/". Текст, находящийся между началом контейнера и его окончанием, является его содержимым.

Примером одиночного тега служит тег перевода на новую строку:
 (браузерами не обрабатываются повледовательности пробелов, символы текстовых редакторов перевода каретки CR и перехода на новую строку LF).

В HTML-документе в наименованиях тегов и их параметров, а также в словах, являющимися их значениями, заглавные и строчные буквы не различаются.



Web-страница помещается в контейнер **<HTML></HTML>** и.состоит из двух частей: заголовка и отображаемого в браузере содержания.

Заголовок страницы помещается в контейнер **<HEAD></HEAD>.** Заголовок содержит название страницы, которое помещается в контейнер **<TITLE></TITLE>** и при просмотре отображается в верхней строке окна браузера.

Также в заголовок помещаются не отображаемые при просмотре **мета-теги**, задающие кодировку страницы для ее правильного отображения в браузере, а также содержащие описание и ключевые слова страницы, которые в первую очередь просматривают роботы поисковых систем.

Мета-теги. В раздел заголовка Web-страницы могут быть добавлены информационные одиночные теги <META>, имеющие атрибуты NAME, HTTP-EQUIV и CONTENT.

Мета тег может информировать браузер о кодировке Web-страницы:

<meta charset=”windows-1251">

Мета теги используются поисковыми системами для индексирования содержания, ключевых слов и автора Web-страницы:

<meta name="Description" content="">

<meta name="Keywords" content="">

<meta name="Author" content="">

Отображаемое в браузере содержание страницы помещается в контейнер **<BODY></BODY>.**

Все создаваемые файлы должны иметь только латинские имена, без использования символов пробелов и спецсимволов. Файлы должны иметь расширения htm или html.

Все файлы, относящиеся к одному проекту: html-файлы, графика, flash-ролики должны располагаться в одном каталоге.

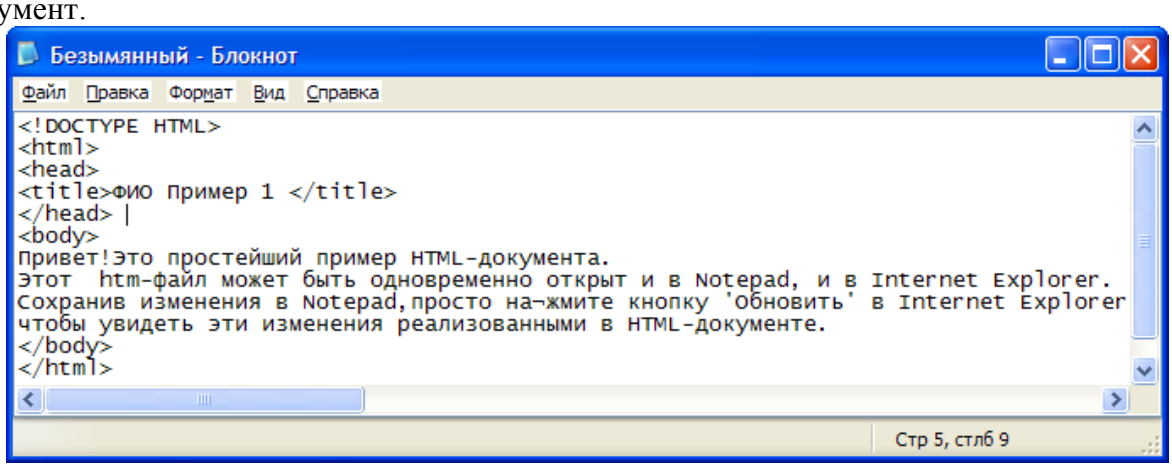
Доктайп

<!DOCTYPE> предназначен для указания типа текущего документа — DTD (document type definition, описание типа документа) для того, чтобы браузер понимал, с какой версией HTML он имеет дело. Если доктайп не указан, браузеры переходят в режим совместимости, в котором не работают многие возможности HTML5, а также возникают ошибки с отображением документа.

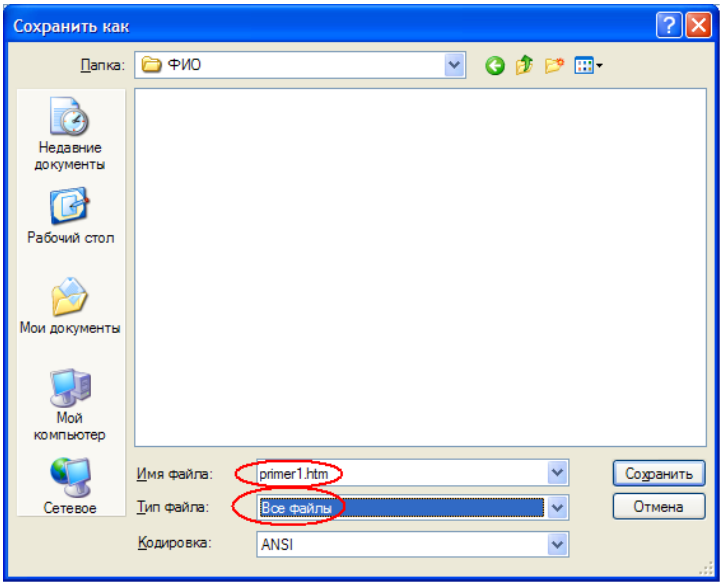
Доктайп не чувствителен к регистру и содержит всего два слова (для HTML5):
<!DOCTYPE html>
Это ключевой элемент и обычно он располагается в первой строке кода.

Пример. Создание простейшей веб-страницы

Откройте текстовый редактор, например БЛОКНОТ. Создайте простой HTML-документ.



При сохранении файла укажите тип файла - все файлы (чтобы программа автоматически не добавила свое расширение), к имени файла добавьте расширение .htm (см. рисунок)



Как видно из примера, вся информация о форматировании документа сосредоточена в его фрагментах, заключенных между знаками "<" и ">". Такой фрагмент (например, <html>) называется меткой (по-английски — tag, читается "тег"). Большинство HTML-меток — парные, то есть на каждую открывающую метку вида <tag> есть закрывающая метка вида </tag> с тем же именем, но с добавлением "/". Метки можно вводить как большими, так и маленькими буквами.

Новые Элементы Разметки в HTML5

Новые элементы для улучшения структуры:

Тег	Описание
<article> (статья)	Определяет независимый, самодостаточный контент: это может быть новостная статья, пост блога, форума или другие статьи, которые могут распространяться независимо от остального содержимого сайта
<aside> (отступление)	Для контента - отступления от темы, в которой он размещен. Отступление должно быть связано с окружающим контентом
<command>	Кнопка, или переключатель (радио кнопка), или флажок

(команда)	
<details> (детали)	Для описания подробностей документа или частей документа
<summary> (резюме)	Заголовок (краткое содержание, резюме) внутри элемента details
<figure> (иллюстрация, рисунок)	Для группировки отдельно стоящего содержимого, может быть видео
<figcaption> (заголовок)	Заголовок раздела иллюстрации, рисунка, схемы и т.п.
<footer> (нижний колонтитул, подвал)	Для создания нижнего колонтитула документа или раздела, может включать, например, имя автора, число, когда документ был опубликован, контактную информацию, информацию об авторском праве
<header> (заголовок)	Для создания вступительной части документа или раздела, может включать панель навигации
<hgroup>	Для группы заголовков с использованием <h1> - <h6>, где наибольший - это главный заголовок раздела, а остальные - это подзаголовки
<mark>	Для текста, который необходимо выделить
<meter>	Для измерения, используется только если максимальное и минимальное значения известны
<nav>	Для панели навигации
<progress>	Прогресс состояния работы
<rt>	Для объяснения ruby-комментария
<rp>	Что показывать браузерам, которые не поддерживают элемент ruby
<section>	Для раздела (колонки, рубрики) в документе. Используется для создания глав, заголовков, нижних колонтитулов или других разделов документа
<time> (время)	Для определения времени или даты, или того и другого
<wbr>	Разделение слов. Для определения возможности разделения слова с целью переноса на другую строку посредством дефиса.

В **HTML 5** для каждой части страницы имеется свой элемент.

Вот их список и краткое назначение:

1) **section**. Назначение - определение секций. Его используют для описания определённого блока текста, например, хорошим применением этого элемента будет при разбиении большой части текста на более малые, как разбиение одной статьи на несколько абзацев.

2) **header**. Назначение - определение верхней секции на странице.

3) **footer**. Назначение - определение нижней секции на странице.

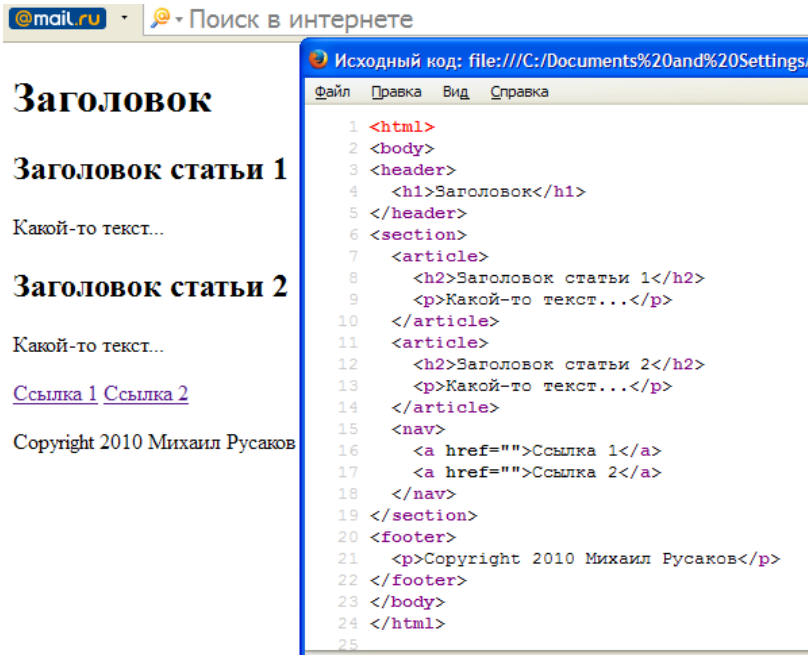
4) **nav**. Назначение - определяет набор ссылок на другие страницы (часто используют для навигации по сайту).

5) **article**. Назначение - выделить определённую часть текста.

Преимущество использования новых элементов следующее: структура документа становится более понятной и ясной не только для Web-мастера, но и для поисковых систем.

Пример. Использование новых элементов разметки HTML5

Сначала определили заголовок в теге **<header>**. Затем добавили секцию с помощью тега **<section>**, внутри которого написали два тега **<article>**, которые выделяют определённую часть текста. После секции поставили набор ссылок с помощью тега **<nav>**.



С помощью тега ** (Не поддерживаются в HTML 5)** можно изменить параметры шрифта. Для тега используются следующие параметры: face, size и color.

Параметр **Face** служит для задания гарнитуры шрифтов использующихся для текста. Названий шрифтов можно указать несколько, через запятую. В этом случае, если первый указанный шрифт не будет найден, будет использоваться следующий по списку.

Пример. Использование параметра face

Текст будет написан шрифтом Arial.

Size задает размер шрифта в условных единицах от 1 до 7. Средний размер, используемый по умолчанию принят 3. Размер шрифта можно указывать как абсолютной величиной (например, size=4), так и относительной (например, size=+1, size=-1). В последнем случае размер изменяется относительно базового.

Color определяет цвет текста, который можно задавать с помощью названий цветов или в шестнадцатеричном формате.

Пример. Изменение цвета текста Первая буква этого предложения будет написана шрифтом Arial, красным цветом и увеличенной.

В таблице перечислены основные теги, которые применяются для изменения оформления текста.

Код HTML	**Описание**	**Пример**
Текст	Жирный текст	**Текст**
Текст	Курсивное начертание текста	*Текст*
<u>Текст</u> (**Не поддерживаются в HTML 5**)	Подчеркнутый текст	Текст
^{Текст}	Верхний индекс	$e=mc^2$
_{Текст}	Нижний индекс	H_2O
<strike>Текст</strike> (**Не поддерживаются в HTML 5**)	Зачеркнутый текст	~~Текст~~
<pre>Текст</pre>	Текст пишется как есть, включая все пробелы	Текст
Текст	Курсивный текст	*Текст*
Текст	Жирный текст	**Текст**

Выравнивание текста. Переход на новую строку. Уровни заголовков.

Выравнивание текста определяет его внешний вид и ориентацию краев абзаца и может выполняться по левому, правому краю, по центру или по ширине.

Код HTML	**Описание**
Текст	Новый параграф, по умолчанию выровненный по левому краю. Перед параграфом автоматически добавляется пустая строка.
Текст	Выравнивание по левому краю.
Текст	Выравнивание по правому краю.
Текст	Выравнивание по центру.
Текст	Выравнивание по ширине.
 	Переход на новую строку
<h1>Текст</h1>…<h6>Текст</ h6>	Теги задания уровней заголовков

Гиперссылки

Для создания ссылки необходимо сообщить браузеру, что является ссылкой, а также указать адрес документа, на который следует сделать ссылку. Оба действия выполняются с помощью тега A, который имеет единственный параметр href. В качестве значения используется адрес документа (URL).

Гиперссылки можно разделить на абсолютные, относительные и внутренние.

Абсолютные ссылки используются в том случае, если мы хотим перейти на документ, адрес которого неизменен Это, прежде всего, ссылки на внешние интернет-ресурсы, например, данный фрагмент

Спросить у Яндекса

Для внутренних ресурсов необходимо использовать **относительные ссылки**. В них дорога к искомому документу считается относительно от того места, где вы сейчас находитесь, например:

** в начало .**

Внутренние ссылки используются для быстрого перемещения внутри документа. Для таких перемещений, во-первых, нужно создать внутри html-документа «**именованные якоря**».

Схема внутренних ссылок.

Они помечаются следующим фрагментом кода: **.**

Чтобы совершить переход в произвольном месте страницы, нужно добавить следующую ссылку: **ссылка.** Знак решетка, сообщит браузеру, что переход совершается внутри страницы. Можно, также, делать ссылку на закладку,

находящуюся в другой веб-странице и даже другом сайте. Для этого в адресе ссылки надлежит указать ее адрес и в конце добавить символ решетки # и имя закладки.

Нумерованные списки

Нумерованные списки представляют собой набор элементов с их порядковыми номерами. Вид и тип нумерации зависит от параметров тега OL, который и используется для создания списка. В качестве маркеров могут быть следующие значения:

- арабские цифры
- заглавные латинские буквы
- прописные латинские буквы
- заглавные римские цифры
- прописные римские цифры

Ниже, в таблице приведены различные параметры тега OL и результат их применения.

Код HTML	**Пример**
  - текст - текст - текст 	Нумерованный список с параметрами по умолчанию: 1. текст 2. текст 3. текст
<ol start="5">	Нумерованный список начинающийся с пяти: 5. текст 6. текст 7. текст
<ol type="A">	Нумерованный список с заглавными буквами латинского алфавита: A. текст B. текст C. текст
<ol type="a">	Нумерованный список с прописными буквами латинского алфавита: a. текст b. текст c. текст
<ol type="I">	Нумерованный список с римскими буквами: I. текст II. текст III. текст
<ol type="i">	Нумерованный список с прописными римскими буквами: i. текст ii. текст iii. текст
<ol type="1">	Нумерованный список с арабскими цифрами: 1. текст 2. текст 3. текст

Маркированные списки

Маркированные списки позволяют разбить большой текст на отдельные блоки. Тем самым привлекается внимание читателя к тексту и повышается его читабельность. С учетом худшего восприятия текста с экрана монитора, чем печатного варианта, это является весьма полезным приемом.

Для установки маркированного списка используется тег UL и LI

Маркеры могут принимать один из трех видов: круг (по умолчанию), окружность и квадрат. Для выбора типа маркера используется параметр type=... тега UL. Вместо многоточия подставляется одно из трех значений указанных в таблице.

Код HTML	**Пример**
<ul type="disc">	Что следует учитывать при тестировании сайта: • работоспособность всех ссылок • поддержку разных браузеров • читабельность текста
<ul type="circle">	Что следует учитывать при тестировании сайта: ○ работоспособность всех ссылок ○ поддержку разных браузеров ○ читабельность текста
<ul type="square">	Что следует учитывать при тестировании сайта: ▪ работоспособность всех ссылок ▪ поддержку разных браузеров ▪ читабельность текста

Таблицы – очень важный элемент web-страниц, они выполняют две функции. Первая - оформление табличной информации: различные расписания, графики дежурств, результаты экспериментов и т.п. А второе - позиционирование элементов страницы.

Таблица состоит из строк и столбцов ячеек, которые могут содержать текст и рисунки. Обычно таблицы используются для упорядочения и представления данных, однако возможности таблиц этим не ограничиваются. С помощью таблиц удобно верстать макеты страниц, расположив нужным образом фрагменты текста и изображений.

Для добавления таблицы на веб-страницу используется тег-контейнер TABLE. Таблица должна содержать хотя бы одну строку и колонку.

Для добавления строк используются теги <tr> и </tr>. Чтобы разделить строки на колонки применяются теги <td> и </td>.

Параметры таблицы

Для изменения вида и свойств таблицы используется множество параметров, которые добавляются в теге TABLE.

<table параметр1=... параметр2=...>

Описание параметров таблицы и их свойств описано ниже.

Атрибут	**Значение**	**Описание**	**Пример**
align=	left right center	Выравнивание таблицы	align=center
background=	URL	Фоновый рисунок	background=pic.gif
bgcolor=	#rrggbb	Цвет фона таблицы	bgcolor=#FF9900
border=	n	Толщина рамки в пикселах	border=2
bordercolor=	#rrggbb	Цвет рамки	bordercolor=#333333
bordercolordark=	#rrggbb	Тень рамки	bordercolordark=#f0f0f0
cellpadding=	n	Расстояние между ячейкой и ее содержимым	cellpadding=7
cellspacing=	n	Дистанция между ячейками	cellspacing=3
nowrap		Запрещает переносы строк в тексте	<table nowrap>
frame=	void	Задание типа рамки	frame=hsides

	above below lhs rhs hsides vsides box	таблицы	
valign=	top bottom	Выравнивание по высоте	valign=top
width=	n n%	Минимальная ширина таблицы, можно задавать в пикселах или процентах	width=90%
height	n n%	Минимальная высота таблицы, можно задавать в пикселах или процентах	height=18

Пример создания таблицы

```
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Добыча</title>
 </head>
 <body>
  <table width="100%" border="1" cellpadding="4">
   <caption>Добыча драгоценных камней</caption>
   <tr>
    <td></td>
    <th>2010</th>
    <th>2012</th>
    <th>2014</th>
   </tr>
   <tr>
    <td>Сапфиры</td>
    <td rowspan="2">1483</td>
    <td>4532</td>
    <td>6743</td>
   </tr>
   <tr>
    <td>Рубины</td>
    <td>3835</td>
    <td>7482</td>
   </tr>
   <tr>
    <td>Алмазы</td>
    <td>683</td>
    <td colspan="2" align="center">238</td>
   </tr>
  </table>
 </body>
</html>
```

Результат:

Добыча драгоценных камней

	2010	2012	2014
Сапфиры	1483	4532	6743
Рубины		3835	7482
Алмазы	683	238	

Вставка изображений.

Для встраивания изображения в документ используется тег IMG, имеющий единственный обязательный параметр src, который определяет адрес файла с картинкой.

Для указания адреса изображения можно задавать как абсолютный, так и относительный адрес.

Пример. Вставка изображения в документ

```
<html>
<body>
<img src="http://www.htmlbook/images/sample.gif"> - это абсолютный адрес размещения изображения
<img src="/images/sample.gif"> - адрес размещения изображения относительно корня сайта
<img src="images/sample.gif"> - адрес размещения изображения относительно текущего HTML-документа
</body>
</html>
```

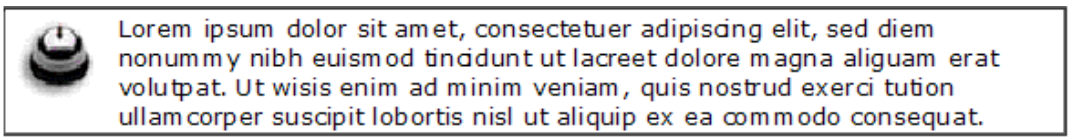
Выравнивание изображений

Способ выравнивания изображений задается параметром align тега IMG.

Наиболее популярные параметры - left и right, создающие обтекание текста вокруг изображения. Чтобы текст не прилегал плотно к рисунку, рекомендуется в теге IMG добавить параметр hspace и vspace, задающих расстояние до текста в пикселах.

Пример. Обтекание текста вокруг рисунка

```
<html>
<body>
<img src="HLPBELL.GIF" width=50 height=50 hspace=10 vspace=10 align=left>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diem nonummy nibh euismod tincidunt ut lacreet dolore magna aliguam erat volutpat. Ut wisis enim ad minim veniam, quis nostrud exerci tution ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.
</body>
</html>
```



Среди атрибутов тега IMG можно выделить и такие:

- width="число" - задает ширину рисунка в пикселях
- height="число" - задает высоту рисунка в пикселях. Для того, чтобы рисунок не искажался, достаточно задать один из атрибутов - ширину или высоту, а второй браузер подберет самостоятельно.
- alt="альтернативный текст" - задает альтернативный текст, который будет написан в пустой рамке рисунка, если по тем или иным причинам рисунок не загрузился, а если рисунок загрузился, то этот надпись будет всплывать при наведении мышью на рисунок.

Создание карт-изображений

Карты - это способ сделать различные части одного графического изображения гиперссылками. Они позволяют выделить отдельные области изображений и определить для каждой из них свое действие.

1. Чтобы создать карту нужно вставить в тег <IMG SRC=""> атрибут USEMAP="#name", где name - имя карты (значок # обязателен).
2. Далее описываются активные области карты. Начините с открывающегося тега <MAP NAME="name"> (здесь повторяется имя, но уже без значка #), а заканчивайте закрывающимся тегом </MAP>.
3. Между этими тегами поместите описание каждой активной области изображения: <AREA SHAPE="форма" COORDS="координаты" HREF="адрес" TITLE="альтернативный текст">.

Элемент <AREA> имеет следующие атрибуты и их значения:

SHAPE	Описывает форму выделяемой области, возможные значения: RECT - прямоугольник CIRCLE - круг POLY - многоугольник DEFAULT - определяет всю область, т.е весь рисунок может стать ссылкой.
COORDS	Координаты, определяющие размеры и положение области на изображении. Все координаты отсчитываются в пикселях от левого верхнего угла изображения. Количество и порядок значений зависит от значения аттрибута SHAPE: RECT: - левый-X, верхний-Y, правый-X, нижний-Y (т.е. сначала координаты левого верхнего угла, затем правого нижнего) CIRCLE: - центр-X, центр-Y, радиус (т.е. горизонтальная и вертикальная координаты центра круга и радиус) POLY: - X1, Y1, X2, Y2, ..., Xn, Yn (просто перечисляются координаты всех вершин многоугольника).
ALT	Альтернативный текст для выделенной области, используется невизуальными браузерами.
TITLE	Название выделенной области, выводится в виде подсказки, всплывающей при наведении курсора на область рисунка.
TARGET	Значение этого аттрибута ("_top", "_blank", "_self" или "_parent") определяет, в каком окне будет открыт документ

Для того, чтобы рассчитать точно координаты нужной части изображения существуют специальные программы. Одна из них называется MapEdit.

Пример создания карты-изображения

```
<img src="images/stories/karta-Maxi.png"
width="600" height="600" alt="Карта" usemap="#mymap" />
<map name="mymap">
  <area shape="rect" coords="0,0,82,126" alt="Участок карты 1" href="1.htm" />
  <area shape="circle" coords="90,58,3" alt="Участок карты 2" href="2.htm" />
  <area shape="circle" coords="124,58,8" alt="Участок карты 3" href="3.htm" />
</map>
```

Новые Медиа Элементы

HTML5 обеспечивает новый стандарт для медиа содержимого:

Тег	Описание
<audio>	Для мультимедийного контента, звуков, музыки или других аудио потоков
<video>	Для видео контента, например, фильмов, клипов или других видео потоков
<source>	Медиа источники ресурсов, определенных в аудио или видео элементах
<embed>	Для встроенного контента, например плагинов

Видео в HTML5

Прежде не было единого стандарта для демонстрации видео на веб странице.

Сегодня большинство видеоклипов воспроизводятся посредством плагинов (например, flash). Однако, не все браузеры имеют одни и те же плагины.

HTML5 определяет стандартный способ включения видео - с помощью элемента video.

Видео Форматы

Формат	IE	Firefox	Opera	Chrome	Safari
Ogg	Нет	3.5+	10.5+	5.0+	Нет
MPEG 4	9.0+	Нет	Нет	5.0+	3.0+
WebM	Нет	4.0+	10.6+	6.0+	Нет

Пример

<video src="movie.ogg" width="320" height="240" controls="controls">
Ваш браузер не поддерживает тег video.</video>

Элемент <video> допускает несколько элементов источников source. Элементы источники могут указывать на различные видео файлы.

Атрибут	Значение	Описание
audio	muted	Определение состояния аудио по умолчанию. На данный момент, допустимо только "muted"
autoplay	autoplay	Если указан, то видео начнет воспроизводиться как только будет готово к воспроизведению
controls	controls	Если указан, кнопки управления (кнопка play и другие) будут показаны
height	пикселы	Устанавливает высоту видеоплеера
loop	loop	Если указан, то видео будет воспроизводиться снова и снова
poster	адресurl	Указывает адрес изображения, представляющего видеоклип
preload	preload	Если указан, видео будет загружаться при загрузке страницы и сразу готово к запуску. Игнорируется, если "autoplay" присутствует
src	адресurl	Адрес URL видеоклипа для воспроизведения
width	пикселы	Установка ширины видео плеера

Аудио в HTML5

Прежде не было единого стандарта для проигрывания аудио в интернете.

Сегодня, большинство аудиотреков проигрываются через плагины (например, flash). Однако, не во всех браузерах установлены одни и те же плагины.

HTML5 Определяет общий способ включения аудио - с помощью элемента audio.

Элемент audio может воспроизводить звуковые файлы, или аудио потоки.

Аудио Форматы

Формат	IE 9	Firefox 3.5	Opera 10.5	Chrome 3.0	Safari 3.0
OggVorbis	Нет	Да	Да	Да	Нет
MP3	Да	Нет	Нет	Да	Да
Wav	Нет	Да	Да	Да	Да

Пример

<audio src="song.ogg" controls="controls"> Браузер не поддерживает элемент audio
</audio>

Элемент <audio> допускает несколько элементов источников source. Элементы источники могут указывать на различные аудио файлы.

<audio controls="controls">
<source src="song.ogg" type="audio/ogg" />
<source src="song.mp3" type="audio/mpeg" /> Браузер не поддерживает элемент audio.
</audio>

Атрибут	Значение	Описание
autoplay	autoplay	Указывает, что аудио начнет проигрываться, как только будет к этому готово.
controls	controls	Отображает кнопки управления, такие как play.
loop	loop	Аудиобудетпроигрыватьсяциклически
preload	preload	Указывает, что аудио должно загружаться вместе с загрузкой веб страницы. Игнорируется, еслиприсутствуетautoplay.
src	адресurl	Определяет адрес аудио, которое будет

		проигрываться

Escape последовательности (символьные объекты)

Некоторые символы являются управляющими символами в HTML и не могут напрямую использоваться в документе. Например, web-браузер воспримет ошибочно следующие символы, добавленные в текст:

- левая угловая скобка "<"
- правая угловая скобка ">"
- амперсанд "&"
- двойные кавычки """ и др.

Кроме того, есть целый ряд символов, которые невозможно набрать с клавиатуры. Например:

- цент "¢"
- знак copyright "©" и др.

Чтобы правильно отобразить подобные символы на html-странице, необходимо заменить их в html-коде escape-последовательностями (их еще называют спецсимволами html). Ниже в таблице приведен список часто встречающихся спецсимволов html:

Символ	**Обычное имя (имена) символа**	**HTML запись символа (escape последовательноть)**
<	символ "меньше чем", левая угловая скобка	<
>	символ "больше чем", правая угловая скобка	>
&	амперсанд	&
"	двойные кавычки	"
©	знак копирайт	©
®	знак зарегистрированной торговой марки	®
	непрерывный пробел	

Указания по выполнению

1. Создайте html-страницу заполнив ее произвольным контентом. Используйте сайт автоматической генерации текста, например, http://online-generators.ru/text
2. Примените новые элементы разметки HTML5, а также элементы разметки HTML4.01 для задания структуры документа. Например, так:

```
<body>
 <header>
   <h1>Название сайта</h1>
   <p>Описание сайта</p>

   <nav>
    <a href='#'>Главная</a>
    <a href='#'>Статьи</a>
    <a href='#'>Новости</a>
    <a href='#'>Ссылки</a>
    <a href='#'>Контакты</a>
   </nav>
  </header>

  <article>
   <header>
    <h2>Название статьи</h2>
    <p>Краткое описание о чем статья</p>

    <figure>
     <img src='http://codingcraft.ru/images/articles/resources/HTML/html5.png' title='HTML5'>
     <figcaption>Логотип HTML5</figcaption>
    </figure>
   </header>

   <section>
    <h3>Глава 1. Название</h3>
    <p>Основной текст...</p>
   </section>

   <section>
    <h3>Глава 2. Название</h3>
    <p>Основной текст...</p>
   </section>

   <section>
    <h3>Комментарии</h3>

    <article>
     <p>Текст комментария</p>

     <footer>
      <small>Опубликовано [автор комментария], [время публикации комментария]</small>
     </footer>
    </article>
   </section>

   <footer>
    <p>Всего комментариев: [xxx], Опубликовано [автор статьи], [время публикации статьи]</p>
   </footer>
  </article>

  <aside>
   Рекламный блок
  </aside>

  <footer>
   <p align='center'>Название ресурса. Все права защищены © 2012.</p>

   <nav>
    <a href='#'>Сходная по тематике статья 1</a>
    <a href='#'>Сходная по тематике статья 2</a>
   </nav>
  </footer>

 </body>
```

3. Создайте на странице ссылки на популярные ресурсы. Примените маркированные и нумерованные списки для группирования ссылок. Все ссылки должны быть сгруппированы в три раздела:

- web-сервисы (ссылки на почтовые, поисковые сервера, чат и т.п.);
- внутренние ресурсы (ссылки на учебники, статьи, скаченные из сети интернет);

- любимые сайты (ссылки на ваши любимые ресурсы сети).

4. Оформите внешний вид страницы, разработайте цветовую схему. Сохраните ваш документ под именем index.htm, сделайте эту страницу – стартовой

5. Добавьте к своей стартовой странице раздел **фотоальбом**, в котором разместите несколько (не более пяти) ваших любимых фотографий. Все фотографии должны быть двух видов: большие размером 600 на 400 пикселей и маленькие размером 120 на 90 пикселей. Уменьшение фотографий выполните в любом известном вам редакторе.

6. Сохраните фотографии в папке с файлом index.htm, дав им однотипные имена: 1.jpg, 2 jpg, для маленьких, и 1_big.jpg, 2_big.jpg для больших. Маленькие фотографии добавьте в файл index.htm.

7. Сделайте каждую маленькую фотографию ссылкой, то есть при щелчке на ней она должна открывать новое окно с увеличенной копией фотографии. Для этого она должна иметь следующий код:

```
<a href="1_big.jpg" target="_new">
<img src="1.jpg" border="0" width="120" height="90" alt="Увеличение по щелчку">
</a>
```

Обратите внимание, что тег <a> имеет параметр **target="_new"**>. Благодаря нему увеличенная фотография откроется в новом окне.

8. Добавьте фон к странице. Будьте осторожны с фоновыми картинками. Во-первых, они должны быть невелики по размеру, чтобы страница быстро загружалась. Во-вторых, они должны иметь свойство текстуры, то есть при выкладывании таких картинок не должно быть видно швов, рисунки должны плавно перетекать друг в друга. И, в третьих, помните, что читать текст поверх текстур очень неудобно.

9. Найдите в сети интернет фоновые рисунки (например, на сайте http://www.woweb.ru/index/0-3) обратите внимание, что каждый файл - это небольшой рисунок, который может продолжаться в стороны, создавая фоновое изображение «без швов».

10. Добавьте для тега <body> следующий параметр (неактуален для HTML5): **<body background="img.gif">.** Фоновый рисунок должен располагаться в той же папке, что и файл index.htm и называться img.gif.

11. Проведите эксперимент, как будет выглядеть ваша страница, насколько удобно читать текст с разными фоновыми изображениями

12. Создайте страницу, содержащую следующий код:

```
<table width="300" border="1" cellpadding="0" cellspacing="0">
<tr bgcolor="#999999">
<td width="100"> </td>
<td colspan="2" align="center">группа 1 </td>
</tr>
<tr>
<td  width="100" rowspan="2" valign="middle">группа 2 </td>
<td  width="100" align="center">1</td>
<td  width="100" align="center">2</td>
</tr>
<tr>
<td width="100" align="center">3</td>
<td width="100" align="center">4</td>
</tr>
</table>
```

13. Проведите исследования «Что произойдет с таблицей, если:

a) изменение ширины

Измените ширину только одной ячейки в колонке. Например, во второй ячейке второй строки увеличьте параметр width= «200». Изменится ли ширина всей колонки?

А что произойдет, если уменьшить значение параметра width= «50». Уменьшится ли ширина всей колонки? Как вы объясните происходящее смещение цифры «1»?

b) таблицы «фиксированные» и «резиновые»

В приведенном примере сумма ширины каждой колонки равна ширине таблицы, а что произойдет, если ширину таблицы сделать больше? Измените в первой строке параметр на width= «500».

Сделайте вашу таблицу «резиновой», измените в первой строке значение параметра width= «300» на width= «50%». Посмотрите, что будет происходить с таблицей при изменении ширины экрана браузера.

с) добавление фонового изображения

Добавьте для первой строки вашей таблицы фоновый рисунок. Во второй строке добавьте параметр background = “fon.gif” (где fon.gif имя вашего фонового рисунка). Удивительно, что фоновый рисунок у первой строки не появился, но еще более удивительно, что если мы откроем полученный html-документ в визуальном редакторе то в нем мы фон увидим!

Для того чтобы фон был виден в браузере, параметр background = “fon.gif” должен задаваться в каждой ячейке строки (строки 3 и 4 в нашем примере).

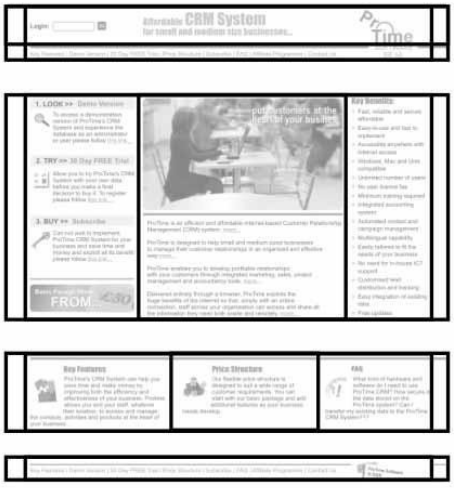
14. Примените полученные знания для создания более сложного вида вашей стартовой страницы. Разбейте ссылки по колонкам:

Стартовая страница с ссылками, разбитыми на колонки.

15. Пользуясь новыми элементами HTML5, разместите на стартовой странице аудио- и видео-файлы.

16. Создайте табличный каркас для данных дизайн-макетов:

Выделение табличного каркаса в дизайн-макете

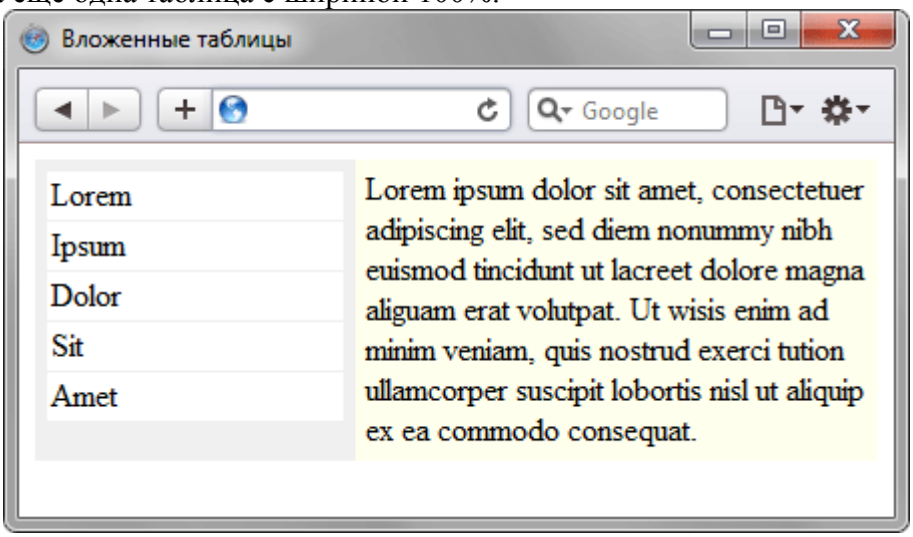


Каркас страницы представлен из четырех таблиц.

Заполните страницы произвольным контентом. Используйте сайт автоматической генерации текста, например, http://online-generators.ru/text.

17. Создаете карту-изображения в новом html-документе. В соответствии с номером компьютера выберите изображение из директории variant для реализации проекта. Задайте произвольные области изображения и определите для них некоторые действия. Щелкнув на активную область изображения, должен осуществляться переход на некоторую страницу, где будет присутствовать текст – «Вы выбрали -…».

18. Создайте макет страницы с помощью вложенных таблиц. Суть идеи проста — в ячейку вкладывается еще одна таблица со своими параметрами. Поскольку эти таблицы в каком-то смысле независимы, то можно создавать довольно причудливые конструкции. Чтобы вложенная таблица занимала всю ширину ячейки, таблице надо задать ширину 100%. В данном макете с помощью таблицы создается две колонки, причем левая колонка имеет фиксированную ширину 150 пикселов. Чтобы создать подобие навигации, внутрь ячейки добавлена еще одна таблица с шириной 100%.



Содержание отчета

1. Цель.
2. Сруктура HTML-документа.

3. Новые элементы разметки HTML5.
4. Создание списков: теги и атрибуты.
5. Создание таблиц: теги и атрибуты.
6. Вставка изображения, аудио и видео-файлов.
7. Выводы

Контрольные вопросы

1. Структура HTML-документа. Перечислите теги заголовка документа.
2. Почему важно делать описание страницы в метатегах?
3. Теги создания абзаца, новой строки, горизонтальной линии.
4. Создание маркированного списка.
5. Создание нумерованного списка.
6. Теги форматирования текста, атрибуты и их значения.
7. Задание фона, фонового изображения на странице.
8. Теги создания таблицы. Использование таблиц для макетирования страницы.
9. Атрибуты тега TABLE.
10. Атрибуты тега TR.
11. Атрибуты тега TD.
12. Создание сложных вложенных таблиц.
13. Использование таблиц для макетирования страницы.
14. Задание гиперссылок на web-странице.
15. Новые элементы для улучшения структуры в HTML5.
16. Перечислить все атрибуты и их назначения тега вставки изображения
17. Какими тегами определяется информация о гиперссылках и о формах активных участков при создании карты изображения.
18. Для чего используется атрибут usemap..
19. Для чего используется атрибут shape..
20. Какие значения может принимать атрибут shape.

Лабораторная работа №№ 4-5
Создание формы на html-странице

Цель: *Получить практические навыки создания html-страницы; научиться создавать веб-формы, выполнять размещение элементов формы с помощью таблиц; выработать навык художественно-эстетического вкуса при оформлении Web-документа*

Ход работы:

7. Изучить теоретический материал.
8. Проработать примеры.
9. Выполнить задания в соответствии с указаниями.
10. Предъявить преподавателю результаты работы: проект и исходный код
11. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
12. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Формы

Форма - это часть HTML-документа, заключенная между парными тегами <FORM> в контейнер.

HTML-документ может содержать в себе несколько форм, однако формы не должны находиться одна внутри другой. HTML-текст, включая метки, может размещаться внутри форм без ограничений. Форма может содержать внутри себя почти все обычные элементы HTML-страницы и, кроме того, - специфические элементы, которые можно помещать только внутри форм: одно- и многострочные поля ввода, флажки, радиокнопки, списки.

Все формы начинаются тегом <FORM> и звершаются тегом </FORM>.

<FORM METHOD="get|post" ACTION="URL"ENCTYPE=”MIME-тип”>
Элементы_формы_и_другие_элементы_HTML
</FORM>

METHOD Метод посылки сообщения с данными из формы. В зависимости от используемого метода вы можете посылать результаты ввода данных в форму двумя путями:

- **GET**: Информация из формы добавляется в конец URL, который был указан в описании заголовка формы.
- **POST**: Данный метод передает всю информацию о форме немедленно после обращения к указанному URL. Ваша CGI-программа получает данные из формы в стандартный поток ввода.

ACTION описывает URL, который будет вызываться для обработки формы. Данный URL почти всегда указывает на CGI-программу, обрабатывающую данную форму. Параметр **ACTION** является обязательным параметром тега**<FORM>:<FORM ACTION="/cgi-bin/form.cgi">**

Впрочем, вместо адреса программы-обработчика значением параметра **ACTION** может быть обычный адрес электронной почты. Тогда данные формы будут просто отправлены по этому адресу, а что с ними делать дальше — решит получатель.

ENCTYPE Тип кодирования данных, введенных через форму, определяется параметром ENCTYPE. Кодирование осуществляется браузером и используется для предотвращения разного рода искажений в процессе передачи на сервер.

Возможными значениями параметра могут быть: **application/x-www-form-urlencoded**(по умолчанию) и **multipart/form-data**. Первое значение используется, если помимо текста необходимо передать на сервер данные иного типа (к примеру, графику или запакованные файлы).

Формат записи состоит из указания типа и его подтипа. ***Тип данных*** — это определение общего типа данных (текст, графика, архив, программа и т. д.), например, **text, image, application**. ***Подтип***— это вид данных внутри определенного общего типа (**image/gif, text/html**).

NAME Параметр NAME присваивает HTML-форме уникальное имя, которое используется в программе-обработчике для идентификации пользовательских данных.

Внутри контейнера <FORM> могут находиться любые теги, кроме другого контейнера <FORM>. Для задания интерфейсных элементов внутри <FORM> используются теги **<input>**, **<select>** и **<textarea>**.

Тег **<INPUT>** используется для задания простого элемента ввода, при этом могут быть использованы следующие атрибуты.

name="идентификатор" - имя для поля ввода, используется при идентификации после подтверждения формы.

type="параметр" - задает тип элемента, параметр может принимать одно из следующих значений:

hidden - Тогда пользователю не предлагаются поля для ввода, но содержимое команды передается при подтверждении и посылке формы;

image - Определяется рисунок, по которому можно сделать щелчок мышью, что приводит к немедленному подтверждению и отсылке формы;

text - Поле ввода текста. Это значение используется по умолчанию;

password - Поле ввода пароля, причем вводимые символы отображаются звездочками;

checkbox - Переключатель, принимающий положения "включено" и "выключено";

radio - Переключатель, принимающий положения "включено" и "выключено", причем может быть выбран только один из элементов

file - Это элемент выбора файла, расположенного на локальном компьютере пользователя, предназначенного для загрузки на Web-сервер.

submit - Кнопка, действие которой сводится к отсылке содержимого заполненной формы на сервер;

reset - Кнопка, которая устанавливает во всех интерфейсных элементах значения по умолчанию.

У атрибута **`type=" "`** в языке HTML5, добавилось следующие значения:

`color` - создаёт виджет выбора цвета,

`date`- содаёт поле для ввода даты, при помощи виджета календаря *(далее ВК)*,

`datetime` - содаёт поле для ввода даты и времени *(ВК)*,

`datetime-local` - содаёт поле для ввода даты и местного времени *(ВК)*,

`email`- создаёт поле для ввода e-mail,

`month` - создаёт поле для ввода месяца *(ВК)*,

`number` - создаёт поле для ввода числа,

`range` - создаёт ползунок,

`search` - создаёт поисковое поле,

`tel` - создаёт поле для ввода телефонного номера,

`time`- создаёт поле для ввода даты и времени *(ВК)*, без часового пояса,

`url` - создаёт поле для ввода url,

`week` - создаёт поле для ввода недели *(ВК)*.

value="значение" - для полей ввода текста или пароля может быть использовано для задания начального содержания поля, а для или задает значение, когда элемент находится в отмеченном состоянии.

size="n" - Задает физический размер поля ввода в символах, однако этот атрибут действует только для элементов ввода текста или пароля.

maxlength="n" - Определяет максимальное количество введенных символов, которые будут приниматься для ввода.

Тег <SELECT> предназначен для создания списков в форме, при этом внутри разрешена только последовательность тегов <OPTION>, за каждым из которых следует некоторое количество простого текста. Атрибуты следующие:

name="идентификатор" - символьное имя для элемента <SELECT>, по которому он идентифицируется;

size="n" - Если значение равно 1 или если этот атрибут опущен, то элемент <SELECT> будет представлен как выпадающее меню; если size=2 или более, то элемент будет представлен как окно выбора, а значение будет определять количество видимых элементов списка.

multiple - Если этот атрибут присутствует, то допускается множественный выбор из списка. Каждый элемент <INPUT> должен включать атрибут NAME=[имя], определяющий имя элемента (и, соответственно, имя переменной, которая будет передана обработчику). Имя должно задаваться **только латинскими буквами**. Большинство элементов <INPUT> должны включать атрибут VALUE="[значение]", определяющий значение, которое будет передано обработчику под этим именем. Для элементов <INPUT TYPE=text> и <INPUT TYPE=password>, однако, этот атрибут не обязателен, поскольку значение соответствующей переменной может вводиться пользователем с клавиатуры.

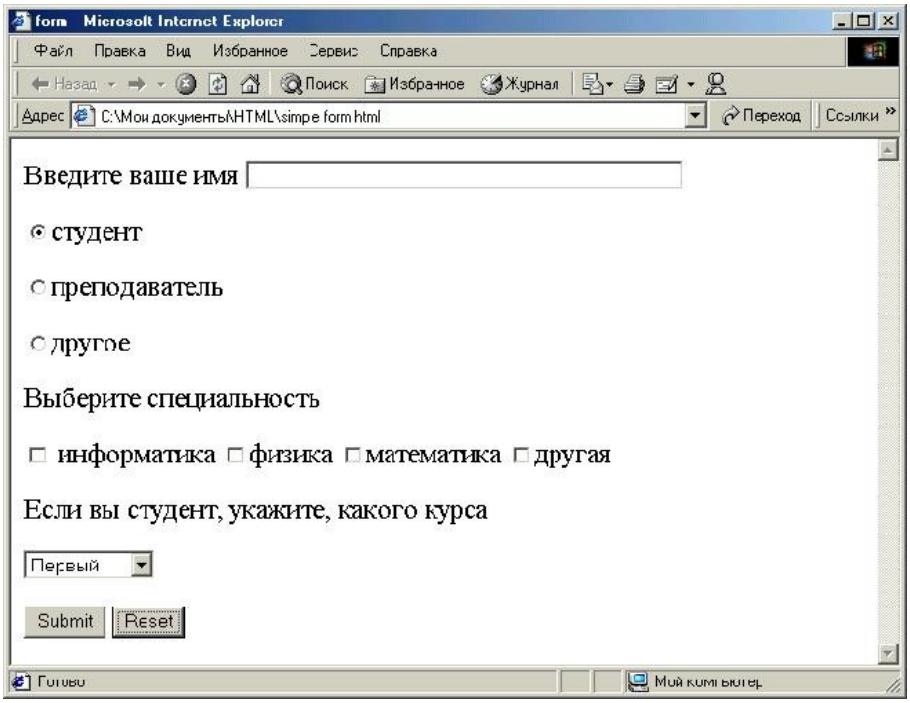
ПРИМЕР 1

```
<FORM method="POST" name="student"
 action=http://www.mycom.com/cgi-bin/script.cgi>
<p>Введите ваше имя <
input type="text" name="t1" size="45"></p>
<p><input type="radio" value="v1" checked name="r1">
студент</p>
<p><input type="radio" value="v2" name="r1">
преподаватель</p>
<p><input type="radio" value="v3" name="r1">
другое</p>
<p>Выберите специальность</p>
<input type=checkbox name="checkbox1">
 информатика</input>
<input type=checkbox name="checkbox2">
физика</input>
<input type=checkbox name="checkbox3">
математика</input>
<input type=checkbox name="checkbox4">
другая</input>

<p>Если вы студент, укажите, какого курса</p>
<select name="select1">
<option value="Первый">Первый
<option value="Второй">Второй
<option value="Третий">Третий
<option value="Четвертый">Четвертый
<option value="Пятый">Пятый
</select>

<p><input type="submit" value="Submit" name="B1">
<input type="reset" value="Reset" name="B2"></p>
</FORM>
```

в результате должно получиться:

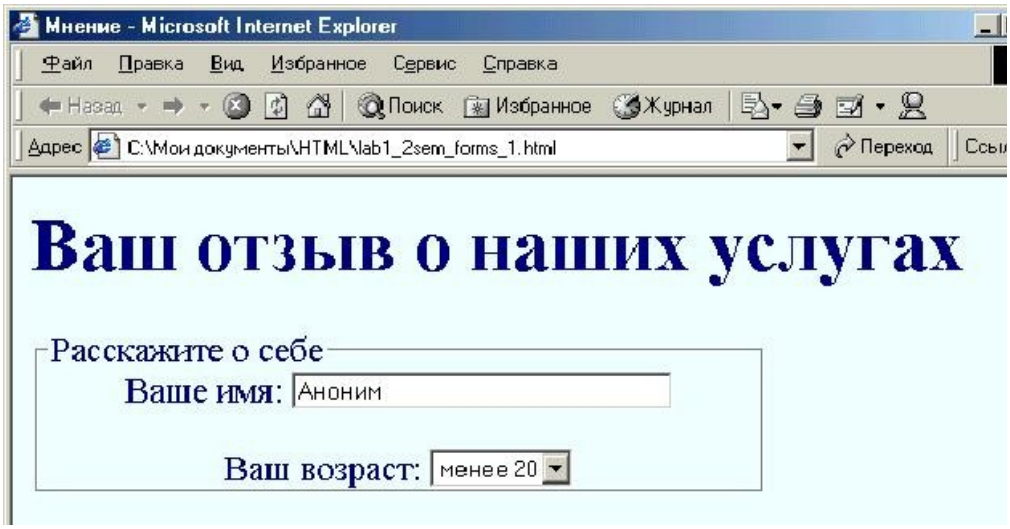


Обрамление вокруг элементов задается с помощью тега <fieldset>

ПРИМЕР 2

```
<fieldset title="Сведения о клиенте">
<legend align="left">Расскажите о себе</legend>
 - задает название блоку
<div align="center">
<label for="Name">Ваше имя:</label>
 - задает текст около текстового поля
<input type="text" name="Name" Value="Аноним"
size="30"><br><br>
<label for="Age">Ваш возраст:</label>
 - задает текст около списка
<select name="Age" size="1">
<option value="10">менее 20
<option value="20">20–30
<option value="30">30–40
<option value="40">40–50
<option value="50">50–60
<option value="60">6олее 60
</select>
</div>
</fieldset>
```

в результате получится фрагмент:



ПРИМЕР 3

```
<HTML>
<HEAD>
<TITLE>Пример 3</TITLE>
</HEAD>
<H1>Несколько более сложная форма </H1>
<FORM ACTION="http://206.31.82.215/hp/nc/fd-win.pht"
 METHOD=post>
<H2>Расскажите немного о себе...</H2>
<P>Указывать подлинные данные совсем не обязательно.
Для целей демонстрации вполне подойдут и вымышленные.
</P>
<P>Имя: <INPUT TYPE=text SIZE=40 NAME=fn><BR>
Фамилия: <INPUT TYPE=text SIZE=40 NAME=ln><BR>
Пол: <INPUT TYPE=radio NAME=gender VALUE="male"
 checked>мужской
<INPUT TYPE=radio NAME=gender
 VALUE="female">женский<BR>
Возраст: <INPUT TYPE=text SIZE=5 NAME=age> лет<BR>
<INPUT TYPE=submit VALUE="Запустить обработчик"></P>
</FORM>
</BODY>
</HTML>
```

Новые атрибуты и их значения в теге **input**:

pattern=" " - указывает на регулярное выражение, исходя из которого принимается решение о верности или не верности введённых данных, атрибут предназначен для создания собственных полей ввода,

Некоторые типовые регулярные выражения **pattern** перечислены в таблице

Выражение	Описание
\d [0-9]	Одна цифра от 0 до 9.
\D [^0-9]	Любой символ кроме цифры.
\s	Пробел.
[A-Z]	Только заглавная латинская буква.
[A-Za-z]	Только латинская буква в любом регистре.
[А-Яа-яЁё]	Только русская буква в любом регистре.
[A-Za-zA-Яа-яЁё]	Любая буква русского и латинского алфавита.
[0-9]{3}	Три цифры.
[A-Za-z]{6,}	Не менее шести латинских букв.

[0-9]{,3}	Не более трёх цифр.
[0-9]{5,10}	От пяти до десяти цифр.
^[a-zA-Z]+$	Любое слово на латинице.
^[А-Яа-яЁё\s]+$	Любое слово на русском включая пробелы.
^[0-9]+$	Любое число.
[0-9]{6}	Почтовый индекс.
\d+(,\d{2})?	Число в формате 1,34 (разделитель запятая).
\d+(\.\d{2})?	Число в формате 2.10 (разделитель точка).
\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}	IP-адрес

ПРИМЕР 4

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Атрибут pattern</title>
 </head>
  <body>
   <form>
    <p>Введите телефон в формате 2-xxx-xxx, где вместо x
    должна быть цифра:</p>
    <p><input type="tel" pattern="2-[0-9]{3}-[0-9]{3}"></p>
    <p><input type="submit" value="Отправить"></p>
   </form>
  </body>
</html>
```

placeholder=" " - указывает на строку, которая находятся в текстовом поле, она исчезает когда пользователь начинает вводить данные,

readonly="readonly" - указывает на текстовое поле, в которое пользователь не может ввести данные,

required="required" - указывает на текстовое поле, которое обязательно должно быть заполненным, если пользователь попытается отправить данные не заполнив это поле, то браузер выдаст об этом сообщение ввиде всплывающей подсказки.

Тег <datalist>

В спецификации **HTML 5** впервые представлены теги <datalist></datalist> . Тег <datalist> определяет список предопределенных вариантов элемента <input>, которые можно выбирать при наборе в текстовом поле.

Теги <datalist></datalist>, с обязательными<option />, не видны в браузере, они лишь определяют допустимые значения при вводе данных посредством элемента <input />. Атрибут list="" тега <input /> указывает на функциональную связь между группой тегов.

Специальные возможности веб-форм

Использование тега LABEL

Тег **<LABEL>**определяет элемент-контейнер, внутрь которого заключается текст подписи к флажку или радиокнопке. Атрибут **for** тега **label** указывает, к какому именно элементу формы относится соответствующая метка. Значение этого атрибута должно совпадать с содержимым атрибута id тега <input>, определяющего нужный флажок или радиокнопку

Группировка элементов форм

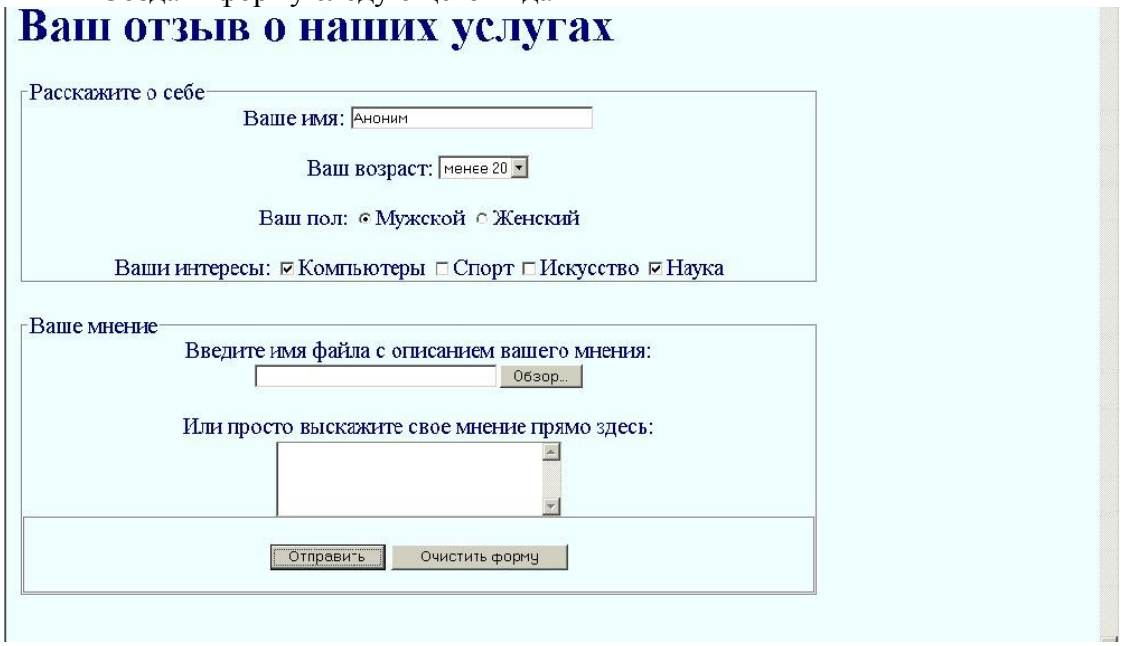
Теги **<FIELDSET>** и **<LEGEND>**, предусмотренные в HTML, позволяют дополнительно структурировать (как логически, так и визуально) сложные формы, разбив их на относительно небольшие и однородные тематические группы элементов.

В случае применения означенных тегов родственные поля веб-формы заключаются в контейнер **<fieldset>.. -</fieldset>**. Элемент **<legend>.. .</legend>**,содержащий текстовый заголовок той или иной группы элементов, должен следовать сразу же за открывающим тегом **<fieldset>**

Указания по выполнению

Внимание! Выполняя задания, используйте возможности HTML5 для оптимизации форм.

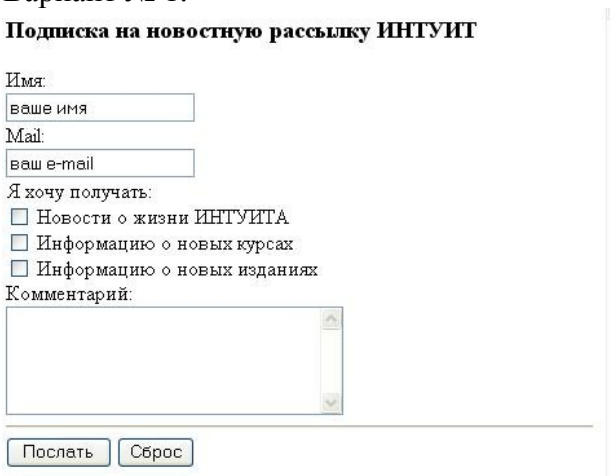
1. Создать форму следующего вида



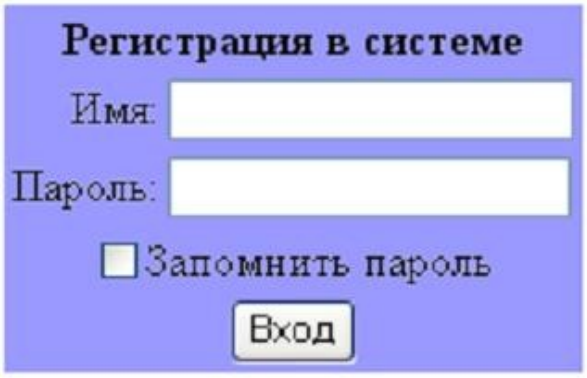
2. Создать форму по вариантам:

Номер компьютера	**1, 6, 11**	**2, 7, 12**	**3, 8, 13**	**4, 9, 14**	**5, 10, 15**
Номер варианта	**1**	**2**	**3**	**4**	**5**

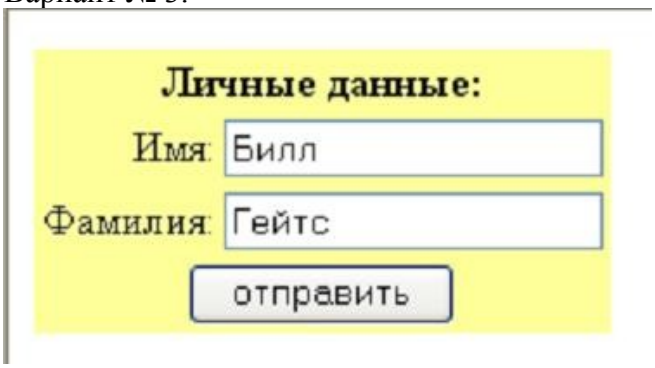
Вариант № 1.



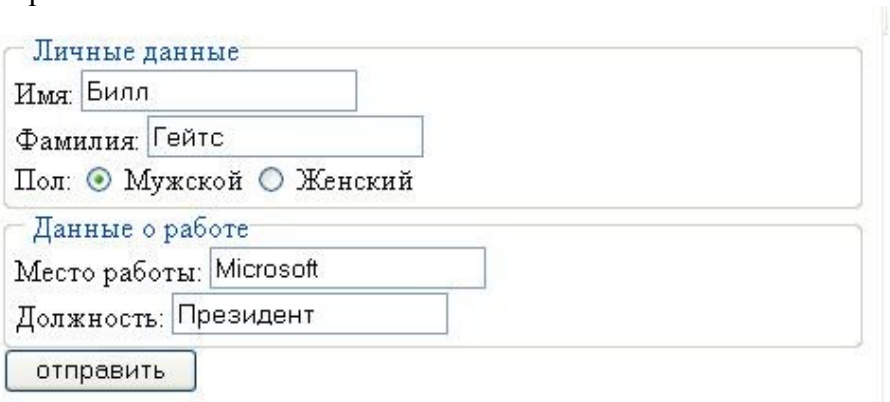
Вариант № 2.



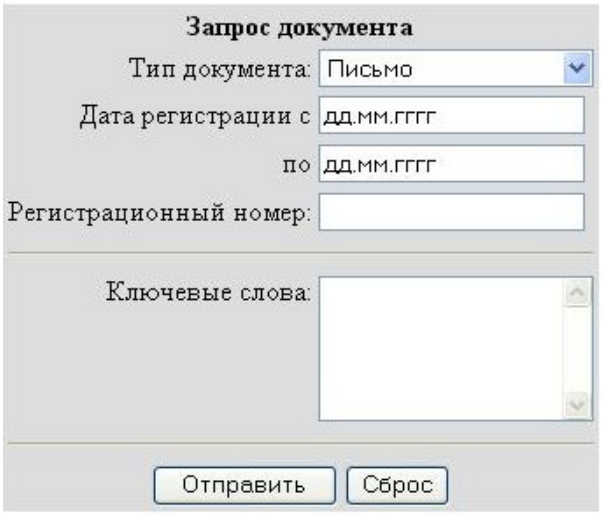
Вариант № 3.



Вариант № 4.



Вариант № 5.



Указания

Для выполнения заданий под номерами 2, 3, 5 необходимо использовать внутри формы таблицы:

```
<form action="/example.php"
method="GET" target="_blank">
<table>
<tr><td>Логин:
<td><input type="text" name="n"/>
<tr>
<td>Пароль:
<td><input type="password" name="p"/>
....
</table></form>
```

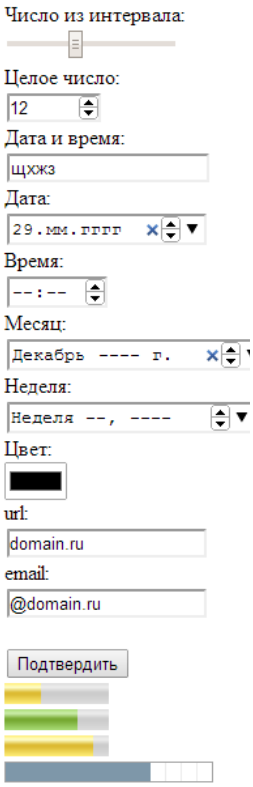
Для выполнения задания под номером 4 задайте обрамление с помощью тега <fieldset>

```
<fieldset>
<legend>Поле авторизации</legend>
Логин: <br /><input type='password' >
.....
</fieldset>
```

3. Создайте форму с использованием HTML5 по образцу:

Исходный HTML код

```
<form>
 Число из интервала:<br/>
 <input type='range' min='0' max='100' step='10' value='10'
 Целое число:<br/>
 <input type='number' min='-100' max='100' step='1' value='
 Дата и время:<br/>
 <input type='datetime'/><br/>
 Дата:<br/>
 <input type='date'/><br/>
 Время:<br/>
 <input type='time'/><br/>
 Месяц:<br/>
 <input type='month'/><br/>
 Неделя:<br/>
 <input type='week'/><br/>
 Цвет:<br/>
 <input type='color'/><br/>
 url:<br/>
 <input type='url' value='domain.ru'/><br/>
 email:<br/>
 <input type='email' value='@domain.ru'/><br/><br/>
 <input type='submit' value='Подтвердить'/>
</form>
```




```
<div><meter min='-50' low='-10' high='30' max='50' value='-15' title='градусы'></meter></div>
<div><meter min='-50' low='-10' high='30' max='50' value='20' optimum='20' title='градусы'></meter></
<div><meter min='-50' low='-10' high='30' max='50' value='35' title='градусы'></meter></div>

<div><progress max='100' value='70' title='%'></progress></div>
```

4. Создайте форму с использованием новых атрибутов и их значений:

Введите email: Не email Отправить

Введите url: Не url Отправить

Введите сотовый телефон (*в формате 8XXXXXXXXXX*): Отправить

Введите домашний телефон (*в формате XXX-XX-XX или XX-XX-XX*): Отправить

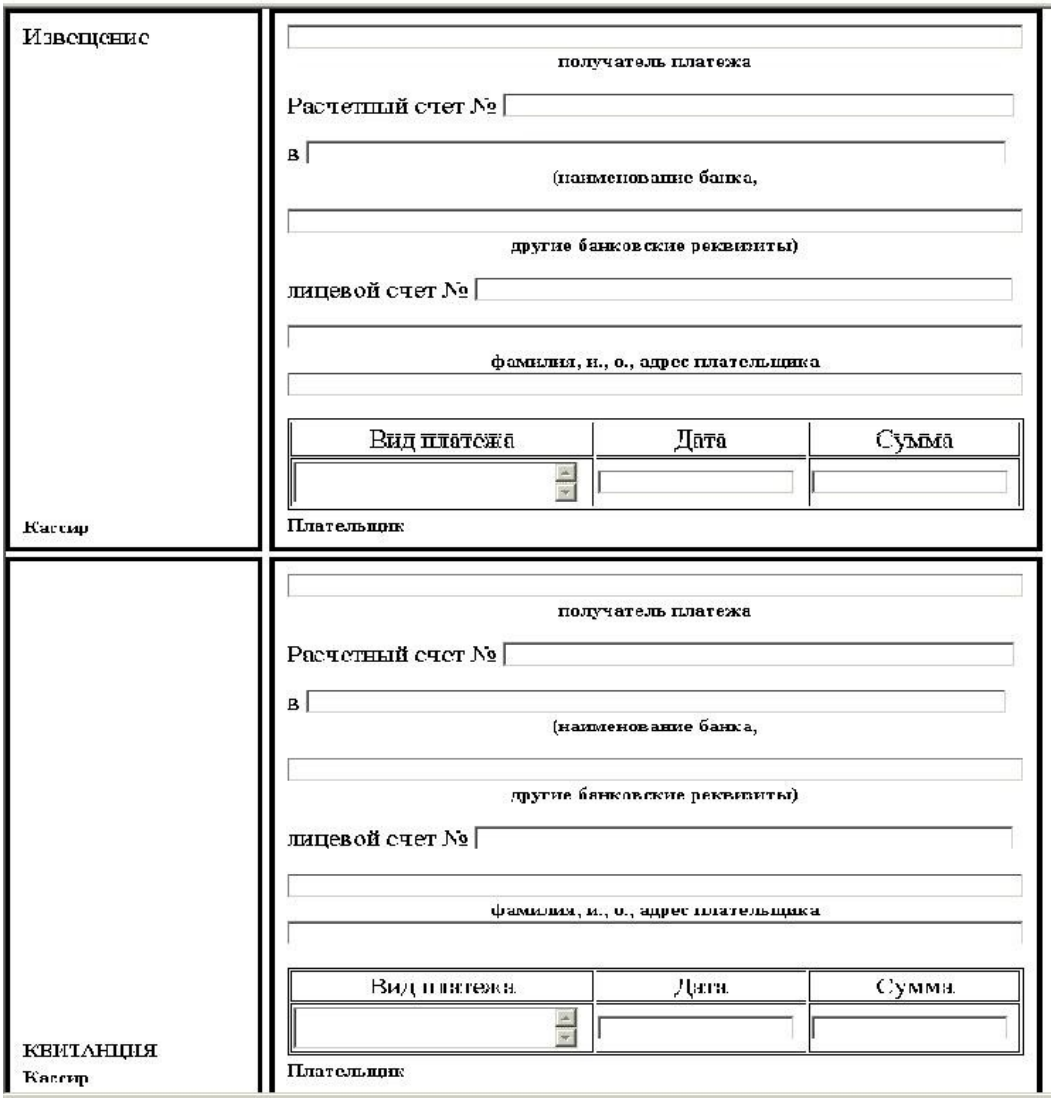
Введите число: 10 Отправить

Выберите число: Отправить

Поиск по сайту Отправить

Выберите цвет: Отправить

5. . Создать электронную форму квитанции. (Необходимо использовать таблицы)



6. Разработать четыре HTML формы в соответствии с таблицами 1 и 2. Дизайн и расположение стандартных управляющих элементов на форме выбираются студентом

самостоятельно. В каждой форме, помимо указанных в таблице 2 элементов должна быть кнопка отправки данных на сервер

Варианты заданий

Таблица 1. Номера HTML-форм, подлежащих разработке

№ варианта	Номера HTML-форм, подлежащих разработке
1	1, 5, 8, 10
2	2, 4, 9, 15
3	1, 7, 10, 16
4	1, 6, 13, 20
5	3, 6, 9, 16
6	5, 14, 15, 17
7	7, 11, 15, 16
8	8, 12, 10, 19
9	6, 8, 12, 13
10	2, 17, 19, 20
11	5, 7, 9, 20
12	3, 8, 13, 19
13	9, 10, 14, 15
14	12, 15, 18, 19
15	6, 10, 13, 15
16	6, 9, 10, 14
17	6, 14, 17, 20
18	1, 9, 15, 16
19	1, 12, 13, 14
20	2, 3, 14, 20
21	6, 7, 9, 16
22	4, 8, 9, 16
23	3, 12, 15, 16
24	12, 14, 16, 19
25	10, 12, 16, 20
26	9, 12, 14, 16
27	3, 14, 18, 19
28	4, 5, 8, 13

Таблица 2. HTML-формы и управляющие элементы

№	Название формы	Метод отправки, адрес сервера	Описание (поля и проверки)
1	Форма авторизации	GET, google.com.ua	Логин (не короче 3 символов, не длиннее 10 символов, только латиница), пароль (не короче 3 символов, не длиннее 10 символов, латиница, цифры, обязательно минимум одна заглавная буква и минимум одна цифра), чекбокс “Запомнить меня”
2	Форма регистрации	POST, yandex.ru	Логин (не короче 5 символов, не длиннее 10 символов, только латиница, только большие буквы), пароль (не короче 3 символов, не длиннее 10 символов, только цифры), фамилия и имя (только русские буквы, первая обязательно заглавная, остальные обязательно строчные),

			чекбокс “Принимаю правила”
3	Форма добавления новости	GET, rozetka.com.ua	Заголовок новости (не короче 10 символов, допускаются русские буквы и латиница, знаки препинания), тело новости (многострочное поле ввода, обязательно не пустое), дата добавления – 3 текстовых поля (обязательно только цифры, год, месяц и день обязательно должны быть корректны)
4	Форма добавления товара в Интернет-магазин	POST, rambler.ru	Заголовок товара (не короче 10 символов, допускаются русские буквы и латиница, знаки препинания), описание товара (многострочное поле ввода, обязательно не пустое), цена товара (текстовое поле, обязательно число с плавающей точкой, в качестве разделителя может быть как точка, так и запятая, обязательно положительное), количество товаров на складе (обязательно целое число, положительное)
5	Форма голосования	GET, ozon.ru	Предмет голосования (обычный текст, без элемента управления) и 5 радиокнопок, содержащих варианты голосования. Обязательно должен быть выбран один вариант.
6	Форма добавления отзыва на товар в интернет магазине	POST, vk.com	Название товара (список одиночного выбора из нескольких позиций), содержание отзыва (многострочное поле ввода), оценка товара – 5 радиокнопок (-2…2). Обязательно должно быть заполнено поле содержания отзыва и выставлена оценка товара.
7	Форма добавления записи в гостевой книге	GET, facebook.com	Автор записи (однострочное текстовое поле), текст записи (многострочное текстовое поле), рубрика (список одиночного выбора). Обязательно должны быть заполнены оба текстовых поля, в поле автор записи должны быть только русские, латинские буквы и пробел.
8	Форма расширенного поиска	POST, samsung.com	Поисковый запрос (однострочное текстовое поле), сайт, на котором должен производиться поиск (однострочное текстовое поле), начальная и конечная даты добавления материала (каждая из дат представляется в виде трех списков одиночного выбора – год, месяц, день). Обязательно должен быть заполнен поисковый запрос, он должен содержать только русские буквы (большие и малые), длина запроса – более 3 символов. В поле сайт должен записываться корректный URL, а обе даты должны иметь верный формат (учитывать проверку на високосный год 29 февраля и количество дней в месяце)
9	Форма обратной связи	GET, microsoft.com	Автор (однострочное текстовое поле), текст сообщения (многострочное текстовое поле), емейл (однострочное текстовое поле), телефон (однострочное текстовое поле). Поля автор и текст сообщения должны быть обязательно

			заполнены. Из полей емейл и телефон может быть заполнено какое-то одно. Если поле заполнено, оно должно быть заполнено корректно (содержать корректный емайл или телефон)
10	Форма бронирования билетов	POST, lg.com	Номер рейса (однострочное текстовое поле), дата поездки (3 списка одиночного выбора – год, месяц и день), время поездки (2 однострочных текстовых поля – час и минута) количество мест (однострочное текстовое поле), сумма (однострочное текстовое поле), способ оплаты (список одиночного выбора). Обязательно все поля должны быть заполнены, номер рейса только целое число, дата поездки должна быть корректной (учитывать високосный год 29 февраля и количество дней в месяце), время поездки – должно быть корректным, количество мест – целое положительное число, больше 0, сумма – положительное вещественное число с округлением только до двух знаков.
11	Форма заказа товара	GET, anekdot.ru	Название товара (однострочное текстовое поле), количество позиций (однострочное текстовое поле), способ оплаты (список одиночного выбора), адрес доставки (однострочное текстовое поле). Обязательно все поля должны быть заполнены, количество позиций – обязательно целое положительное число, большее нуля.
12	Форма вызова специалиста на дом	POST, wikipedia.org	Описание проблемы (многострочное текстовое поле), домашний адрес (однострочное текстовое поле), желаемые дата (3 списка одиночного выбора – год, месяц, день) и время (2 однострочных текстовых поля – часы, минуты) прихода специалиста, мобильный телефон (однострочное текстовое поле). Обязательно должно быть заполнено описание проблемы и домашний адрес. Остальные поля необязательны, но если они заполнены, то заполнение должно быть корректным. Так при вводе даты нужно учитывать високосный год 29 февраля и количество дней в месяце, при вводе времени – минуты должны задаваться в диапазоне от 0 до 59, часы – от 8 до 16.
13	Форма анкетирования	GET, php.net	Вопрос анкеты (обычный текст, без элемента управления) и 4 чекбокса, содержащих варианты ответов. Обязательно должен быть выбран хотя бы один вариант.
14	Форма бронирования номера в гостинице	POST, drupal.ru	Количество мест (-одно, -двух, -трех, -четырех местный номер – список одиночного выбора), уровень (3, 4, 5 звезд, люкс – список одиночного выбора), ФИО постояльца (3 однострочных текстовых поля), начальная дата бронирования (3 списка одиночного выбора – год, месяц, день), количество суток, способ оплаты (список

			одиночного выбора). Обязательно должны быть заполнены ФИО, начальная дата бронирования, количество суток. При вводе даты нужно проверять високосный год 29 февраля и количество дней в зависимости от выбранного месяца
15	Форма вопрос-ответ	GET, nvidia.com	Содержание вопроса (многострочное поле ввода), рубрика вопроса (список одиночного выбора). Содержание вопроса не должно быть пустым и должно содержать только латиницу, русские буквы и знаки препинания (.,;: круглые скобки и пробел)
16	Форма добавления комментария	POST, alawar.ru	Автор комментария (однострочное текстовое поле), текст комментария (многострочное поле ввода), при вводе комментария должно появляться сообщение «у Вас осталось 1000 символов». Более 1000 символов вводить не допускается. Оба поля не должны быть пустыми
17	Форма добавления сообщения в чате	GET, electrolux.com	Автор сообщения (однострочное текстовое поле), тип сообщения (приватное, публичное - чекбокс), адресат сообщения (однострочное текстовое поле), текст сообщения (многострочное поле ввода). Обязательно должен быть указан автор сообщения, а если тип сообщения – приватное, то обязательно должен быть указан и адресат. Текст сообщения не должен быть пустым.
18	Форма добавления ip адреса	POST, price.ua	4 однострочных текстовых поля, задающих компоненты IP адреса v4. Каждая компонента – обязательно целое число от 0 до 255. Ввод локальных и зарезервированных значений не допускается. К примеру, 127.0.0.1, 10.x.x.x и т.п. являются зарезервированными значениями. Полный перечень зарезервированных IP адресов есть в Википедии.
19	Форма добавления контактов	GET, youtube.com	Имя пользователя (однострочное текстовое поле), домашний адрес (однострочное текстовое поле), телефон (однострочное текстовое поле), скайп (однострочное текстовое поле), ссылка на страницу в соцсети (однострочное текстовое поле), емайл (однострочное текстовое поле). Имя пользователя задается обязательно, остальные поля по желанию, но хотя бы одно из оставшихся полей должно быть заполнено. Если поле заполнено – должна быть минимальная проверка его на корректность.
20	Форма записи в клинику	POST, ex.ua	Описание недомогания (многострочное поле ввода), предполагаемый специалист (список одиночного выбора – лор, окулист, терапевт и т.п.), желаемые дата и время посещения, результаты анализа (поле выбора файла). При вводе даты нужно учитывать високосный год 29 февраля и количество дней в месяце, при вводе

			времени – минуты должны задаваться в диапазоне от 0 до 59, часы – от 8 до 16. Описание недомогания обязательно должно быть заполнено. Результаты анализа прикреплять необязательно.

Содержание отчета

8. Цель.
9. Создание формы.
10. Основные теги формы, их назначение
11. Новые атрибуты и значения
12. Выводы

Контрольные вопросы

1. Синтаксис формы
2. Атрибуты METHOD и ACTION. Их возможные значения.
3. Тег TEXTAREA
4. Тег INPUT
5. Атрибуты тега INPUT: CHECKED, MAXLENGTH, NAME, SIZE, SRC, TYPE, RESET SUBMIT, TEXT, RADIO, PASSWORD, CHECKBOX
6. Значия атрибута TYPE HTML5
7. Каким образомреализуется меню выбора в формах
8. Для чего предназначен тип элемента INPUT под названием File
9. Как отправить форму почтой?
10. С помощью какого тега задается обрамление вокруг элементов формы?

Лабораторная работа №№ 6-7-8

Форматирование web-страниц с использованием каскадных таблиц стилей

Цель: *Получить практические навыки использования каскадных таблиц стилей для задания внешнего вида Web-страниц.*

Ход работы:

1. Изучить теоретический материал.
2. Выполнить задания в соответствии с указаниями.
3. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
4. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Стили, а правильнее каскадные таблицы стилей (хотя иногда их называют и листами стилей), а еще правильней – *Cascading Style CSS* (сокращенно *CSS*) – это набор шаблонов или стилей, которые применяются к какой-то части документа, или вообще ко всему документу, с помощью которых браузер отображает содержимое документа в том виде, в котором ему предписывают стили.

С помощью CSS можно определять стиль и вид текста. Аналогично тому, что используется тег FONT, задающий свойства шрифта, но стили обладают большими возможностями и позволяют сократить код HTML.

Правописание стилей

Селекторы (Selectors):

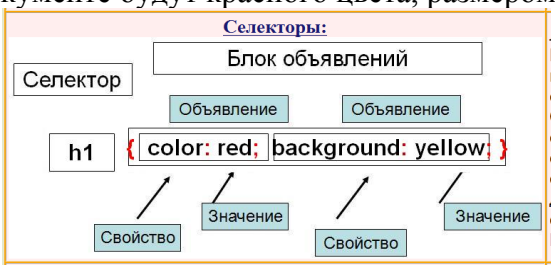
Синтаксисис: **селектор {свойства}**

Любой элемент HTML - это возможный CSS селектор. Свойства селектора определяют стиль элемента, для которого он определен.

Пример:

```
H1 {color:red; size:20pt;}
```

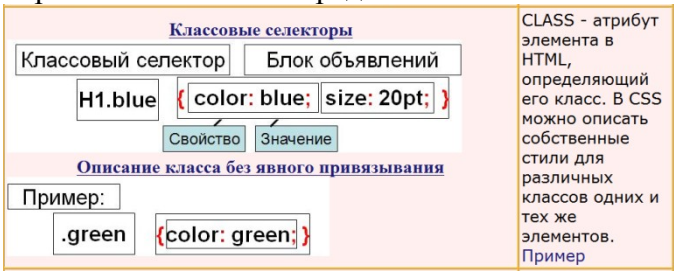
Все элементы H1 в документе будут красного цвета, размером в 20 точек (pt, point).



Классовые селекторы:

Синтаксис: **селектор.класс {свойства}**

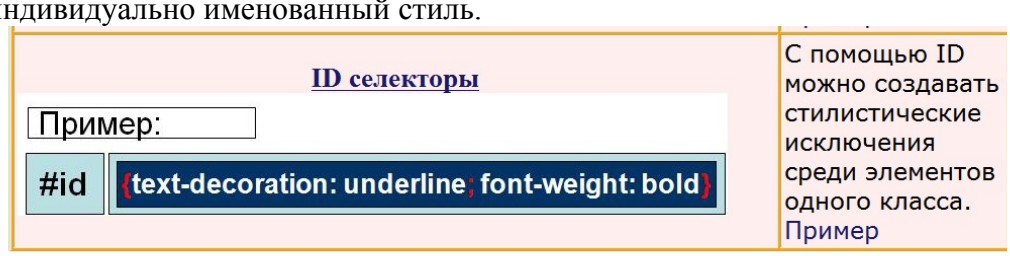
class - атрибут элемента в HTML, определяющий его класс. В CSS можно описать собственные стили для различных классов одних и тех же элементов. Классы могут так же быть описаны без явного привязывания их к определенным элементам.



ID селекторы:

Синтакс: #id {свойства}

id - индивидуально именованный стиль.



id - атрибут элемента в HTML, определяющий его индивидуальный стиль.

Контекстуальные селекторы:

Контекстные селекторы - это сочетания нескольких обыкновенных селекторов. Стиль задается только элеметнам в заданной последовательности в зависимости от каскадного порядка.

Пример:

P EM {color:silver;}

В данном примере все элементы EM внутри элементов P будут иметь заданный стиль.

Придание нескольким элементам одинаковых свойств:

Скажем Вам нужно придать нескольким элементам Вашей веб страницы одинаковых свойств. В этом случае при определении селекторы перечисляются через запятую перед блоком свойств.

ПРИМЕР:

h1,h2,h3,p,strong {color:green; font-style:italic;}

Все элементы h1, h2, h3, p и strong будут зелеными.

Псевдоклассы и псевдоэлементы :

Синтаксис:

селектор:псевдокласс { свойства }

селектор.класс:псевдокласс { свойства }

селектор:псевдоэлемент { свойства }

селектор.класс:псевдоэлемент { свойства }

Псевдоклассы и псевдоэлементы - это особые классы и элементы, присущие CSS и автоматически определяемые поддерживеющими CSS браузерами. Псевдоклассы различают разные типы одного элемента, создавая при определении собственные стили для каждого из них. Псевдоэлементы являются частями других элементов, задавая этим частям отличный от элемента вцелом стиль.

Список псевдоклассов и псевдоэлементов :

Anchor Pseudo Classes - эти псевдоклассы элемента <a href=" ">, обозначающего ссылку. Псевдоклассы этого элемента: (ссылка), active (активная ссылка), visited (посещенный ранее URL), hover (псевдокласс, возникающий при поднесении курсора к ссылке, не работает в Нетскейпе).

First Line Pseudo-element - first-line. Этот псевдоэлемент может быть использован с block-level элементами (p, h1 и т.д.). Он изменяет стиль первой строки этих элементов.

First Letter Pseudo-element - first-letter. Похож на first-line, но влияет не на всю строку, а только на первый символ.

ПРИМЕР :

```
a:link,a:visited {color:blue}
a:active {color:red}
a:hover {text-decoration:none}
```

В данном примере все элементы Anchor (ссылки) будут синими. При нажатии (в активном состоянии) поменяют цвет на красный. И при подведении курсора мышки исчезнет подчеркивание.

1. **Внутренние Таблицы Стилей**

Использование Внутренних стилей мало чем отличается от использования обычных HTML тегов. Они задают стиль только одному элементу документа при помощи атрибута STYLE в HTML теге.

Пример html:

```
<font color="blue" size="3" face="Arial"> Вперед в будущее </font>
```

2. Глобальные Таблицы Стилей

Глобальные стили задают вид элементов всего документа. Для этого используется тег <style type="text/css">. Размещается в заголовке документа.

Пример:

```
<head>
< style type="text/css">
body {background:black; font-size:9pt; color:red; font-family:Arial Black}
.base{color:blue; font-style:italic}
h1 {color:white}
#bold {font-weight:bold}
</ style >
</head>
```

3. Связанные Таблицы Стилей

Связанные таблицы стилей используются для придания нескольким документам одного стиля и хранятся в отдельном файле. Это очень привлекательно, когда нужно выдержать сайт в одном стиле, не утруждая себя составлением таблиц для каждого документа.

Пример:

Файл styles.css

```
1  < style type="text/css">
2  body {background:black; font-size:9pt; color:red; font-family:Arial Black}
3  .base{color:blue; font-style:italic}
4  h1 {color:white}
5  #bold {font-weight:bold}
6  </ style >
7
```

В самих же HTML документах делается ссылка на этот файл при помощи тега <link>. Выглядит это так:

```
<link rel="stylesheet" type="text/css" href="путь до файла">
```

Мы можем задать стиль для каждого тега.

Свойства шрифта

Свойство	Значение	Описание	Пример
font-family	имя шрифта	Задает список шрифтов	p {font-family: Arial, serif}
font-style	normal italic oblique	Нормальный шрифт Курсив Наклонный шрифт	p {font-style: italic}
font-variant	normal small-caps	Капитель (особые прописные буквы)	p {font-variant: small-caps}
font-weight	normal lighter bold bolder 100-900	Нормальная жирность Светлое начертание Полужирный Жирный 100-светлый шрифт, 900-самый жирный	p {font-weight: bold}
font-size	Размер шрифта normal pt px %	нормальный размер пункты пикселы проценты	font-size: normal font-size: 12pt font-size: 12px font-size: 120%

Пример 1. Задание свойств шрифта с помощью CSS

```
<html>
<style>
H1 { font-family: Arial, Helvetica, Verdana, sans-serif; font-size: 150%; font-weight: light }
</style>
<body>
<H1>Заголовок</H1>
Обычный текст
</body>
</html>
```

Пример 2. Использование различных параметров шрифтов

Пример	**Пример**	**Пример**	ПРИМЕР	***Пример***
font-family: Verdana, sans-serif; font-size: 120%; font-weight: light	font-size: large; font-weight: bold	font-family: Arial, sans-serif; font-size: x-small; font-weight: bold	font-variant: small-caps	font-style: italic; font-weight: bold

Свойства текста

Кроме изменения параметров шрифтов, можно управлять и свойствами всего текста. Значения свойств приведены в таблице.

Свойство	**Значение**	**Описание**	**Пример**
line-height	normal множитель точно %	Интерлиньяж (межстрочный интервал)	line-height: normal line-height: 1.5 line-height: 12px line-height: 120%
text-decoration	none underline overline line-through blink	Убрать все оформление Подчеркивание Линия над текстом Перечеркивание Мигание текста	text-decoration: none
text-transform	none capitalize uppercase lowercase	Убрать все эффекты Начинать С Прописных ВСЕ ПРОПИСНЫЕ все строчные	text-transform: capitalize
text-align	left right center justify	Выравнивание текста	text-align: justify выравнивание по ширине
text-indent	точно %	Отступ первой строки	text-indent:15px; text-indent:10%

Пример 3. Использование различных параметров текста

Пример: и это все о нем	Пример: текст по центру	Пример: Это не ссылка, а просто текст	Пример: отступ первой строки	Пример: полуторный межстрочный интервал
text-transform: capitalize	text-align:center	text-decoration: underline	text-indent: 20px	line-height: 1.5

Границы и рамки

Спецификация CSS описывает несколько свойств, с помощью которых можно создавать границу вокруг различных элементов и управлять ее видом. Границы - одна из наиболее слабых сторон CSS, т.к. браузеры содержат большое количество ошибок и по-разному интерпретируют параметры. Старшие версии браузеров отображают рамки вокруг элементов более корректно.

Свойство	Значение	Описание	Пример
padding-top **padding-right** **padding-bottom** **padding-left** **padding**	значение %	Отступ от границы элемента до его содержимого	table {padding: 15px 15px}
border-top-width **border-right-width** **border-bottom-width** **border-left-width** **border-width**	thin medium thick значение	Ширина границы	P {border-top-width: 4px}
border-color	цвет	Цвет границы	P {border-color: red}
border-style	none dotted dashed solid double groove ridge inset outset	Стиль рамки	table {border-style: double}
border-top **border-right** **border-bottom** **border-left**	border-top-width border-style цвет	Определяет толщину, стиль и цвет каждой границы	table {border-top: solid 4px red; border-left: solid 4px blue}
border	см. выше	Задает толщину, стиль и цвет рамки	table {border: solid 4px red}

Пример 4. Типы рамок

Для управления видом рамки предоставляется восемь значений параметра border-style.

```
<p style="color: yellow;background-color: deepskyblue; text-decoration: underline; text-transform: uppercase; border: pink inset 25; PADDING: 20; font-size: larger; line-height: 40px; text-align: center;"> .. </p>
```

В РЕЗУЛЬТАТЕ ПРИМЕНЕНИЯ УКАЗАННЫХ СТИЛЕВЫХ СВОЙСТВ К ДАННОМУ АБЗАЦУ ДОЛЖНО ПОЛУЧИТЬСЯ СЛЕДУЮЩЕЕ

атрибут style="" задает стилевое оформление абзаца

атрибут color: yellow; задает цвет текста

атрибут background-color: deepskyblue; задает цвет фона для абзаца

атрибут text-decoration: underline; задает подчеркивание для текста

атрибут text-transform: uppercase; задает режим заглавных букв для текста

атрибут border: pink inset 25; задает рамку вокруг абзаца, соответственно, розовую выпуклую толщиной 25 пикселей

атрибут font-size: larger; задает размер шрифта

атрибут line-height: 40px; задает межстрочный интервал

атрибут text-align: center; задает выравнивание текста внутри абзаца по центру

Псевдо-классы

:active Добавляет специальный стиль активированному элементу

:focus Добавляет специальный стиль элементу, когда элемент находится в фокусе

:hover Добавляет специальный стиль элементу, когда указатель мыши находится над элементом

:link Добавляет специальный стиль непосещенной ссылке

:visited Добавляет специальный стиль посещенной ссылке

:first-child Добавляет специальный стиль элементу, который является первым потомком некоторого другого элемента

:lang Позволяет автору определить используемый в заданном элементе язык

Указания по выполнению

Задание 1. Откройте блокнот, назовите файл *mystyle.css* и напишите в нем следующее:

```
BODY {background-color:papayawhip}

P {color:maroon}

H2 {color:red}
```

Теперь создайте новую страницу *Familiya.html*:

```
<HTML>
<HEAD>
<TITLE>Пример 1</TITLE>
<LINK REL=STYLESHEET TYPE="text/css" HREF="mystyle.css">
</HEAD>
<BODY>
<H2>Нервные люди</H2>
<P>Недавно в нашей коммунальной квартире драка произошла. И не то что драка, а целый бой. На углу Глазовой и Боровой.
<P>Дрались, конечно, от чистого сердца. Инвалиду Гаврилову последнюю башку чуть не оттяпали.
<P>Главная причина – народ очень уж нервный. Расстраивается по всяким пустякам. Горячится. И через это дерется грубо, как в тумане.
<P>Оно, конечно, после гражданской войны нервы, говорят, у народа завсегда расшатываются. Может оно и так, а только у инвалида Гаврилова от этой идеологии башка поскорее не зарастет.
</BODY>
</HTML>
```

Задание 2. Оцените результат, создайте свой стиль и оформите веб-страницу в новом стиле (еще один файл *.css*).

Задание 3.

Создайте документ *mystyle1.css*:

```
BODY {background-color:#FFEFD5}
P {color:#800000; text-align:right; font-style:italic}
H2 {color:#F00; text-align:center}
```

Обратите внимание, что одно определение отделяется от другого точкой с запятой (*;*), а после последнего определения точка с запятой не ставится.

Сохраните изменения и посмотрите, что получится в файле *Familiya.html*, созданном на прошлой лабораторной работе.

Задание 4.

Запись во внешнем файле - не единственная возможность добавления стилей *CSS* в документ. Можно его ввести в самом документе при помощи атрибута *STYLE=...*. Создайте новый файл *Familiya_2.html*.

```
<HTML>¶
<HEAD>¶
<TITLE>Пример·2</TITLE>¶
</HEAD>¶
<BODY·STYLE="background-color:lightgrey">¶
<H2·STYLE="color:red;·text-align:center">Нервные·люди</H2>¶
<P·STYLE="color:brown;·text-align:right;·font-style:italic">¶
Недавно·в·нашей·коммунальной·квартире·драка·произошла.·
И·не·то·что·драка,·а·целый·бой.·На·углу·Глазовой·и·
Боровой.¶
<P·STYLE="color:blue;·text-align:left">¶
Дрались,·конечно,·от·чистого·сердца.·Инвалиду·Гаврило-
ву·последнюю·башку·чуть·не·оттяпали.¶
<P·STYLE="color:brown;·text-align:right;·font-style:italic">¶
Главная·причина·-·народ·очень·уж·нервный.·
Расстраивается·по·всяким·пустякам.·Горячится.·И·через·
это·дерется·грубо,·как·в·тумане.¶
<P·STYLE="color:blue;·text-align:left">¶
Оно,·конечно,·после·гражданской·войны·нервы,·говорят,·у·
народа·завсегда·расшатываются.·Может·оно·и·так,·а·
только·у·инвалида·Гаврилова·от·этой·идеологии·башка·
поскорее·не·зарастет.¶
</BODY>¶
</HTML>¶
```

Итак, после нужного тега пишем *STYLE=...*, и затем в кавычках даем свойство и затем, после двоеточия – значение этого свойства. В этом случае не нужно никакого внешнего файла.

Задание 5.

Есть еще один метод включения CSS в документ – так называемый глобальный или встроенный стиль. Этот стиль записывается в заголовке страницы, он воздействует на всю страницу.

```
<HTML>
  <HEAD>
    <TITLE>Пример 4</TITLE>
    <STYLE TYPE="text/css">
      BODY {background-color:#000}
      P {color:#FF3; text-align:right; font-style:italic}
      H2 {color:#F03; text-align:center}
    </STYLE>
  </HEAD>
  <BODY>
    <H2>Нервные люди</H2>
    <P>Недавно в нашей коммунальной квартире драка произошла. И не то что драка, а целый
бой.
 На углу Глазовой и Боровой.
    <P>Дрались, конечно, от чистого сердца. Инвалиду Гаврило-ву последнюю башку чуть не
оттяпали.
    <P>Главная причина - народ очень уж нервный. Расстраива-ется по всяким пустякам.
Горячится.
И через это дерется грубо, как в тумане.
    <P>Оно, конечно, после гражданской войны нервы, говорят, у народа завсегда
расшатываются.
 Может оно и так, а только у инвалида Гаврилова от этой идеологии башка поскорее не
зарастет.
  </BODY>
</HTML>
```

Как видите, весь стиль описывается между тегками *</HEAD>* и *</HEAD>*. Он начинается со строчки **<STYLE TYPE="text/css">**, и заканчивается **</STYLE>**. А внутри записываются все правила.

Задание 6. Использование свойств шрифта.

Создайте веб-страницу, задав различные свойства шрифта, по образцу, представленному на рисунке.

ШРИФТАМ В КОМПЬЮТЕРНОЙ ГРАФИКЕ ВСЕГДА УДЕЛЯЛОСЬ МНОГО ВНИМАНИЯ. WORLD WIDE WEB В ЭТОМ АСПЕКТЕ НЕ ЯВЛЯЕТСЯ ИСКЛЮЧЕНИЕМ. НО ВСЕ БОГАТСТВО И РАЗНООБРАЗИЕ ШРИФТОВ, СУЩЕСТВУЮЩИХ В ПРИРОДЕ, ДЛЯ РУССКОГО ЯЗЫКА ОГРАНИЧЕНО ФАКТИЧЕСКИ ТРЕМЯ ШРИФТАМИ: SERIF(ОБЫЧНО TIMES ИЛИ ДРУГОЙ ШРИФТ С ЗАСЕЧКАМИ), SANS-SERIF(ARIAL, ИЛИ HELVETICA, ИЛИ ДРУГОЙ ШРИФТ БЕЗ ЗАСЕЧЕК) И MONOSPACE(COURIER). ЕСЛИ БЫТЬ ТОЧНЫМ, ТО ЗДЕСЬ ПЕРЕЧИСЛЕНЫ СЕМЕЙСТВА ШРИФТОВ. ОБЫЧНО, КАЖДОЕ ИЗ ЭТИХ СЕМЕЙСТВ ПРЕДСТАВЛЕНО ТОЛЬКО ОДНИМ КИРИЛЛИЧЕСКИМ ШРИФТОМ.

АВТОР ДОКУМЕНТА ДЛЯ УПРАВЛЕНИЯ ОТОБРАЖЕНИЕМ БУКВ ТЕКСТА МОЖЕТ ПРИМЕНИТЬ НЕСКОЛЬКО АТРИБУТОВ, ВЛИЯЮЩИХ НА ШРИФТ:

FONT-FAMILY

СЕМЕЙСТВО НАЧЕРТАНИЙ ШРИФТА (ГАРНИТУРА);

FONT-STYLE

ПРЯМОЕ НАЧЕРТАНИЕ ИЛИ КУРСИВ;

FONT-WEIGHT

УСИЛЕНИЕ"(НАСЫЩЕННОСТЬ) ШРИФТА, "ЖИРНОСТЬ" БУКВ;

СУЩЕСТВУЕТ ТАКЖЕ ВОЗМОЖНОСТЬ СОВМЕСТИТЬ ВСЕ ЭТИ ПАРАМЕТРЫ В ОДНОМ АТРИБУТЕ FONT:

FONT:BOLD 12PT SANS;

Все свойства шрифта задайте в одной строке:

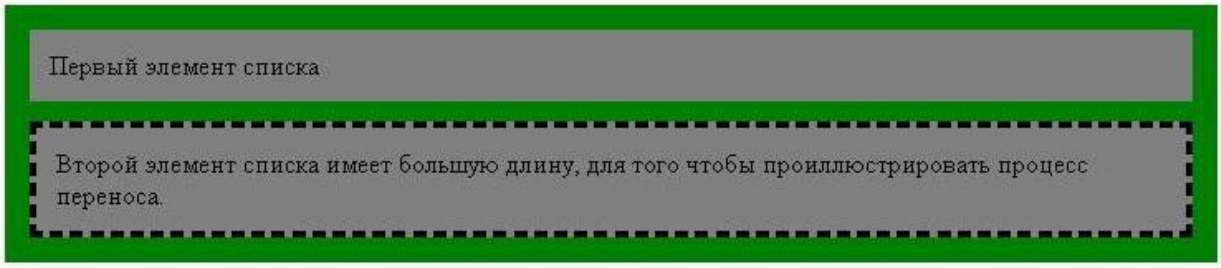
```
<style type="text/css">
p.ex1
{font:15px arial,sans-serif;}
p.ex2
{font:italic bold 12px/30px Georgia,serif;}
</style>
```

Селекторы по классам

Задание 7. Создайте глобальный стиль *CSS* для таблицы.

```
<!DOCTYPE html>
<html>
<head>
<style type="text/css">
.lux {
    width: 300px;
    border: 1px solid #a52a2a;
    border-collapse: collapse;
    border-spacing: 0;
   }
   .lux th {
    background: #a52a2a;
    color: white;
   }
   .lux td {
    border-bottom: 1px solid black;
   }
   .lux td, .lux th {
    padding: 4px;
   }
</style>
</head>
<body>
< <table class="lux">
    <tr><th></th><th>2004</th><th>2005</th><th>2006</th></tr>
    <tr><td>Рубины</td><td>43</td><td>51</td><td>79</td></tr>
    <tr><td>Изумруды</td><td>28</td><td>34</td><td>48</td></tr>
    <tr><td>Сапфиры</td><td>29</td><td>57</td><td>36</td></tr>
    <tr><td>Аметисты</td><td>23</td><td>64</td><td>97</td></tr>
  </table>
</html>
```

Задание 8. Создайте глобальный стиль *CSS* для таблицы по образцу



Селекторы по ID

Отличие селектора ID от селектора по классам лишь в том, что классы можно применять сколько угодно раз, а *ID* – один. Узнать уникальный идентификатор можно по решетке (#), которая ставится перед значением элемента.

Задание 9. Создайте уникальный индентификатор *ID*, который ответственнен за красный цвет.

```
#red {color:red}
h1#blue {color:blue}
```

Задайте два индентификатора *ID*, первый – красного цвета можно использовать с любым элементом, второй же (голубой) может быть применен только к классам заголовка первого уровня. Этот код пишется в разделе *style*.

А в теле документа напишете:

```
<P ID=red> Этот абзац красного цвета
<h1 ID=blue> А этот заголовок первого уровня голубого цвета
```

Контекстные селекторы

Они используются, когда одни теги приходится вкладывать в другие.

Например, нам хочется, чтобы все слова в абзаце, выделенные курсивом были красного цвета. Тогда мы можем написать просто так:

```
P I {color:red}
```

Обратите внимание, что теги в этом случае пишутся через пробел.

Задание 10. Создайте контекстный селектор для выделения красным цветом курсивного начертания в абзаце.

```
<HTML>  <HEAD>
  <TITLE>Контекстные селекторы</TITLE>
 <STYLE TYPE="text/css">
  P I {color:red}
 </STYLE>
 </HEAD>
 <BODY>
  <p>
   Это вот у нас абзац обыкновенный <i>а эти слова написаны красным курсивом</i> а
дальше опять обыкновенный абзац.
  </p>
 </BODY>
</HTML>
```

Это вот у нас абзац обыкновенный *а эти слова написаны красным курсивом* а дальше опять обыкновенный абзац.

Задание 11. Добавьте теперь выделение в абзаце полужирного начертания зеленым (green) цветом, подчеркнутого – синим (blue), курсивного – красным (red), верхнего индекса – цветом maroon, нижнего индекса – цветом magenta.

Для создания этого документа вам не только надо посмотреть только что созданную страницу, но и вспомнить, с помощью каких тегов оформляется **полужирное**, подчеркнутое и *курсивное* начертани текста,а с помощью каких - верхний и нижний индексы.

Псевдоэлементы

Псевдоэлемент *first-letter* отвечает за Первую Букву Абзаца (параграфа), а *first-line* – за Первую Линию Абзаца (параграфа).

Задание 12. Создайте документ с использованием псевдоэлементов, где первая буква предложения – красная, первая строка абзаца – синего цвета в режиме CAPSLOCK.

```
<head>
<style type="text/css">
p:first-letter
{
color:#ff0000;
font-size:xx-large;
}
p:first-line
{
color:#0000ff;
font-variant:small-caps;
}
</style>
</head>
<body>
<p>Вы можете комбинировать псевдо-элементы :first-letter и :first-line,
чтобы добавить специальный эффект к первой букве и первой строке текста!</p>
</body>
```

Псевдоклассы

К псевдоклассам относятся все состояния ссылок.

С помощью подручных средств *CSS* мы можем переопределить цвета ссылок по умолчанию.

Итак, есть четыре состояния ссылок:

- **link** – цвет еще не посещенной ссылки;
- **active** – это когда на ссылку нажали;
- **visited** – цвет посещенной ссылки;
- **hover** – когда только навели мышкой на ссылку/

Задание 13. Изучить псевдоклассы на примере. Изменить цвета ссылок по умолчанию на следующие: #1EE6DF; #6CEAB3; #C5EB14; и #1F4292. Для каждого состояния сделать свой фон. Активные ссылки должны быть полужирными, неподчеркнутыми, оставить подчеркнутыми только наведенные ссылки.

```
<head>
<style type="text/css">
a.one:link {color:#ff0000;}
a.one:visited {color:#0000ff;}
a.one:hover {color:#ffcc00;}
a.two:link {color:#ff0000;}
a.two:visited {color:#0000ff;}
a.two:hover {font-size:150%;}
a.three:link {color:#ff0000;}
a.three:visited {color:#0000ff;}
a.three:hover {background:#66ff66;}
a.four:link {color:#ff0000;}
a.four:visited {color:#0000ff;}
a.four:hover {font-family:monospace;}
a.five:link {color:#ff0000;text-decoration:none;}
a.five:visited {color:#0000ff;text-decoration:none;}
a.five:hover {text-decoration:underline;}
</style>
</head>
<body>
<p>Наведите курсор мыши над ссылками, чтобы увидеть как они меняют оформление.</p>
<p><b><a class="one" href="index.php" target="_blank">Эта ссылка меняет цвет</a></b></p>
<p><b><a class="two" href="index.php" target="_blank">Эта ссылка меняет размер шрифта</a></b></p>
<p><b><a class="three" href="index.php" target="_blank">Эта ссылка меняет цвет фона</a></b></p>
<p><b><a class="four" href="index.php" target="_blank">Эта ссылка меняет семейство шрифта</a></b></p>
<p><b><a class="five" href="index.php" target="_blank">Эта ссылка меняет декорацию текста</a></b></p>
</body>
```

Background

С помощью тега *background* можно установить цвет, положение, изображение, привязку и повторяемость.

Background-color

Цвет можно установить не только для фона всей страницы, а вообще для любого элемента: для абзаца, заголовка и так далее.

Задание 14. Задать фон для заголовка, параграфа и блока.

```
<html>
<head>
<style type="text/css">
h1{background-color:#6495ed;}
p{background-color:#e0ffff;}
div{background-color:#b0c4de;}
</style>
</head>
<body>
<h1>CSS фоновые цвета - пример!</h1>
<div>
Это текст внутри элемента div.
<p>Этот параграф имеет свой собственный цвет фона.</p>
Мы все еще внутри элемента div.
</div>
</body>
</html>
```

Background-image

Для использования в качестве фона картинок, их желательно сохранить в папке image.

Задание 15. Задать фон для текста, списка и абзаца предыдущего примера. В качестве фона использовать изображения (найти в сети).

```
h1{background-image:url('bgdesert1.jpg');}
p{background-image:url('bgdesert2.jpg');}
div {background-image:url('bgdesert3.jpg');}
```

Background-repeat

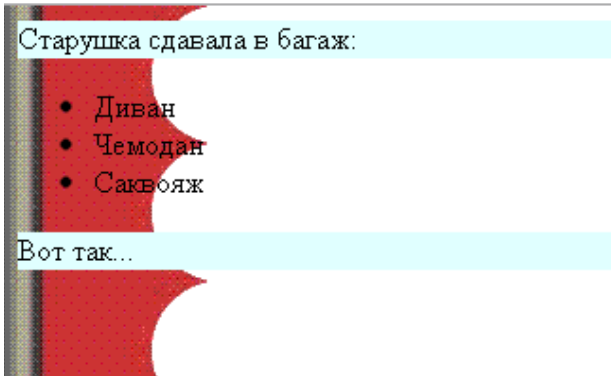
Это свойство определяет, будет ли фоновое изображение заполнять все пространство (*repeat* – по умолчанию), либо не будет повторяться (*no-repeat*), либо повторяется по горизонтали (*repeat-x*), или по вертикали (*repeat-y*).

Задание 16. Задать фон из картинки по вертикали и цветовой фон для параграфа.

Ниже приведен код для описания стиля, однако, имейте ввиду, что у вас могут быть другие имена рисунков.

```
<style type="text/css">
body
{background-image:url('picture.gif');
background-repeat:repeat-y;}
p
{ background-color:#e0ffff; }
</style>
```

Должен быть примерно такой результат



Задание 17. Расположить фоновый рисунок в точно заданном месте с помощью свойства **Background-position**. Оно может быть задано и в числах, и в процентах, и в словосочетаниях *top, center, bottom, left* или *right*.

```
<head>
<style type="text/css">
body
(background-image:url('img_tree.png');
background-repeat:no-repeat;
background-position:right top;
margin-right:200px;)
</style>
</head>
```

Точно также можно расположить и практически любые элементы относительно окна браузера.

Задание 18. Используйте свойства CSS для получения результата:

1: **Green bold text**
2:

Этот параграф отображается белым цветом по черному фону

Эта гипертекстовая ссылка отображается белым цветом по черному фону.
3: Это ссылка меню (красная на белом фоне) *Это ссылка абзаца (цвет морской волны, курсив)*
4: *Основная гипертекстовая ссылка* *Модифицированная гипертекстовая ссылка*
5: Этот текст отображается зеленым цветом И этот текст тоже зеленый Эта часть гиперссылки отображается полужирным перес

Содержание отчета

13. Цель
14. Синтаксис задания CSS
15. Классы, id-стили
16. Способы внедрения CSS
17. Пример использования стилей
18. Выводы

Контрольные вопросы

1. Назначение CSS
2. Способы внедрения CSS
3. Синтаксис
4. Отступы(margin)
5. Границы(border)
6. Цвет текста(color)
7. Цвет фона текста(background-color)
8. Шрифт(font)

Лабораторная работа №№ 9-10-11

Динамические эффекты с использованием CSS

Цель: *Получить практические навыки использования каскадных таблиц стилей для задания позиционирования элементов и блочной верстки Web-страниц.*

Ход работы:

1. Изучить теоретический материал.
2. Проработать примеры из теоретической части.
3. Выполнить задания в соответствии с указаниями.
4. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
5. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Позиционирование элемента

До сих пор мы рассматривали CSS-свойства, предназначенные для оформления HTML-элементов. Мы научились менять цвет текста, фона, делать рамки, выбирать шрифт и так далее. Но при создании сайта важной задачей является размещение HTML-элементов в нужных местах страницы, то есть задача вёрстки.

Ранее мы верстали сайты с помощью таблиц. Если модульная сетка не сложная, то табличная вёрстка является вполне разумным решением. Когда таблиц становится много, то в них очень сложно ориентироваться.

Посмотрите на веб-страницу на рис.1 (сайт rsport.ru). Здесь огромное количество элементов расположены в очень сложной модульной сетке. Такую сетку практически невозможно реализовать с помощью таблиц.

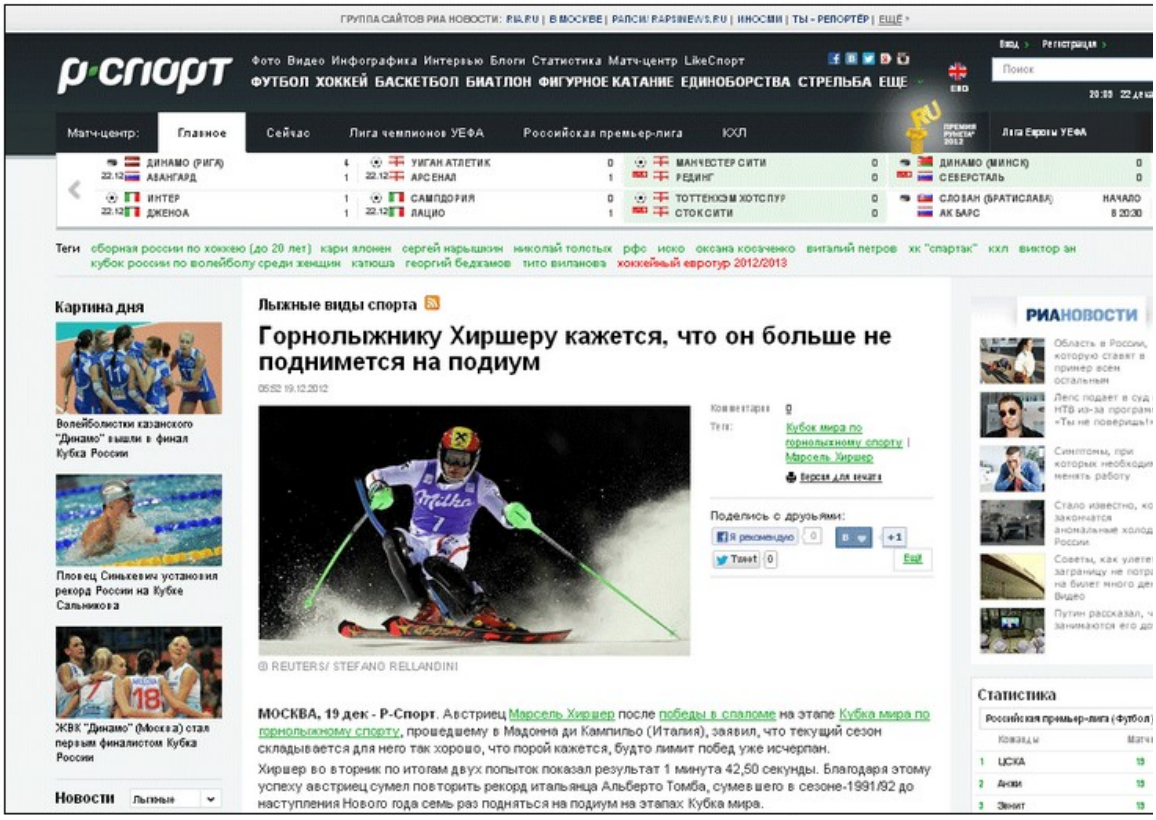


Рис. 1 Пример страницы со сложной вёрсткой

Абсолютное позиционирование элемента

Обычно элемент веб-страницы отображается после предыдущих элементов, записанных в коде HTML-документа. Рассмотрим примеры.

`<p>Текст абзаца <img src="arrow.gif"> </p>`	Текст абзаца

Тег `<img>` в документе записан после текста абзаца, значит и на странице картинка отобразится после текста.

`<p>Воздушный шарик состоит из:</p>`
`<ul>`
`<li>резиновой оболочки;</li>`
`<li>завязки;</li>`
`<li>воздуха.</li>`
`</ul>`
	Воздушный шарик состоит из:
	• резиновой оболочки;
	• завязки;
	• воздуха.

Код `<ul>` записан после `<p>`, поэтому на странице список отобразится ниже параграфа.

Таким образом, элементы на странице по умолчанию никогда не перекрываются. Однако этот порядок можно менять с помощью CSS-свойств `position`, `float` и `clear`.

`position`

CSS-свойство `position` позиционирует элемент независимо от расположения предшествующих элементов. Свойство имеет два важных значения: `absolute` и `relative`.

Элемент со свойством `position:absolute;` располагается независимо от всех других элементов. Координаты этого элемента относительно левого и верхнего краёв страницы определяются свойствами `left` и `top`.

Приведём код CSS-правила для букета, который расположился в левом верхнем углу страницы:

```
position: absolute;
left: 0px;
top: 0px;
```

Как видите, картинка отобразилась в верхнем левом углу независимо от других элементов и перекрывает их в нескольких местах.

Значение `absolute` удобно в том случае, когда элемент нужно “привязать” к углу или краю страницы. При редактировании страницы, добавлении и удалении содержимого, элемент будет оставаться на месте независимо от других.

Плавающий элемент

Так как тег `<p>` является блочным, то он отображается ниже предыдущего элемента с новой строки. В примере ниже записаны обычные теги картинки и двух абзацев.

```
<img src="images/swan.gif" />
<p>Эти грациозные белоснежные птицы во все времена привлекали людей своей
строгой красотой. ... </p>
<p>Но восторг красотой и воспевание лебедей в поэзии не мешали людям охотиться
на этих крупных водоплавающих птиц ...  </p>
```

Результат показан на рис. 2.

Эти грациозные белоснежные птицы во все времена привлекали людей своей строгой красотой. Лебеди с самых ранних веков человеческой истории служили в Европе символом весны и тепла. Их образ являлся олицетворением женской красоты, нежности, романтики, поэзии и верности. В античности лебедь считался птицей Афродиты и Аполлона, а христианская иконография сделала его птицей Девы Марии.

Но восторг красотой и воспевание лебедей в поэзии не мешали людям охотиться на этих крупных водоплавающих птиц ряди мяса, пуха и пера. Так что, к настоящему времени многие виды лебедей находятся под угрозой и занесены в Красную книгу. Сохранению поголовья лебедей помогают на только различные природоохранные мероприятия, но и, ставшее в последнее время распространённым, разведение лебедей в неволе.

Рис. 2 Обычный порядок следования HTML-элементов

Справа от картинки остаётся пустое место, что не красиво. Кроме того неэффективно используется площадь страницы. Необходимо, чтобы текст обтекал картинку справа.

CSS-свойство **`float`**(плавать) прижимает элемент к краю страницы, чтобы остальные могли его «обтекать» с другой стороны. Свойство `float` имеет два важных значения:

- `left` сдвигает элемент к левому краю страницы;
- `right` сдвигает элемент к правому краю страницы.

Пример:

```
.обтекание {
   float: left;
   margin-right: 15px;
}

<img src="images/swan.gif" class="обтекание" />
<p>Эти грациозные белоснежные птицы во все времена привлекали людей своей
строгой красотой. ... </p>
<p>Но восторг красотой и воспевание лебедей в поэзии не мешали людям охотиться
на этих крупных водоплавающих птиц ...  </p>
```

Результат показан на рис. 3.

Эти грациозные белоснежные птицы во все времена привлекали людей своей строгой красотой. Лебеди с самых ранних веков человеческой истории служили в Европе символом весны и тепла. Их образ являлся олицетворением женской красоты, нежности, романтики, поэзии и верности. В античности лебедь считался птицей Афродиты и Аполлона, а христианская иконография сделала его птицей Девы Марии.

Но восторг красотой и воспевание лебедей в поэзии не мешали людям охотиться на этих крупных водоплавающих птиц ряди мяса, пуха и пера. Так что, к настоящему времени многие виды лебедей находятся под угрозой и занесены в Красную книгу. Сохранению поголовья лебедей помогают на только различные природоохранные мероприятия, но и, ставшее в последнее время распространённым, разведение лебедей в неволе.

Рис. 3 Картинка «плавает влево», а абзацы обтекают её

В предыдущем примере (см. рис.3) первый абзац полностью поместился справа от картинки и второй абзац продолжил её обтекание. Если требуется, чтобы второй абзац расположился ниже картинки, то необходимо «досрочное» прекращение обтекания.

CSS-свойство **`clear`** (очищать, освобождать) освобождает от обтекания предыдущих плавающих элементов. Чаще всего используют значение `both`, которое освобождает от обтекания как слева, так и справа. В примере ниже второму абзацу присвоено свойство `clear:both`.

```
.обтекание {
   float: left;
   margin-right: 15px;
}
.ниже {
   clear: both;
}

<img src="images/swan.gif" class="обтекание" />
<p>Эти грациозные белоснежные птицы во все времена привлекали людей своей
строгой красотой. ... </p>
<p class="ниже" >Но восторг красотой и воспевание лебедей в поэзии не мешали
людям охотиться на этих крупных водоплавающих птиц ...  </p>
```

Результат показан на рис. 4.

Эти грациозные белоснежные птицы во все времена привлекали людей своей строгой красотой. Лебеди с самых ранних веков человеческой истории служили в Европе символом весны и тепла. Их образ являлся олицетворением женской красоты, нежности, романтики, поэзии и верности. В античности лебедь считался птицей Афродиты и Аполлона, а христианская иконография сделала его птицей Девы Марии.

Но восторг красотой и воспевание лебедей в поэзии не мешали людям охотиться на этих крупных водоплавающих птиц ряди мяса, пуха и пера. Так что, к настоящему времени многие виды лебедей находятся под угрозой и занесены в Красную книгу. Сохранению поголовья лебедей помогают на только различные природоохранные мероприятия, но и, ставшее в последнее время распространённым, разведение лебедей в неволе.

Рис. 4 Второй абзац прекратил обтекание картинки и расположился ниже с новой строки

position	позиционирование элемента
left	сдвиг относительно левой границы
top	сдвиг относительно верхней границы
float	плавающий элемент
clear	освобождение от плавающего элемента

**Блочная вёрстка с помощью тега **

Как мы уже знаем, `<div></div>` — это служебный тег, служащий контейнером (или упаковкой) для других тегов. Содержимое тега ведёт себя как единое целое, а значит, мы можем вводить для него общее оформление. Внутрь его можно поместить другие элементы (параграфы, картинки, и т. д.). Таким образом получается крупный блок с разнообразным содержимым, который значительно легче позиционировать на веб-странице, нежели каждый из элементов в отдельности. Элемент `<div>` — блочный и ведёт себя как отдельный прямоугольный блок, то есть отображается в отдельной строке.

Сверстаем с помощью блоков веб-страницу с трёхколоночной модульной сеткой. Создадим пять блоков и поместим в них заголовок сайта, навигационное меню, основное содержимое, колонку новостей и сведения об авторских правах.

Создадим первый блок `<div>`, поместим в него заголовок «Сайт студента Васильева В. В.» и отформатируем его как заголовок второго уровня `<h2>`. Поставим текстовый курсор после предыдущего блока и вставим новый `<div>`. Разместим в нём навигационные ссылки. Точно так же создадим блоки новостей и статьи. Результат показан на рис. 1.

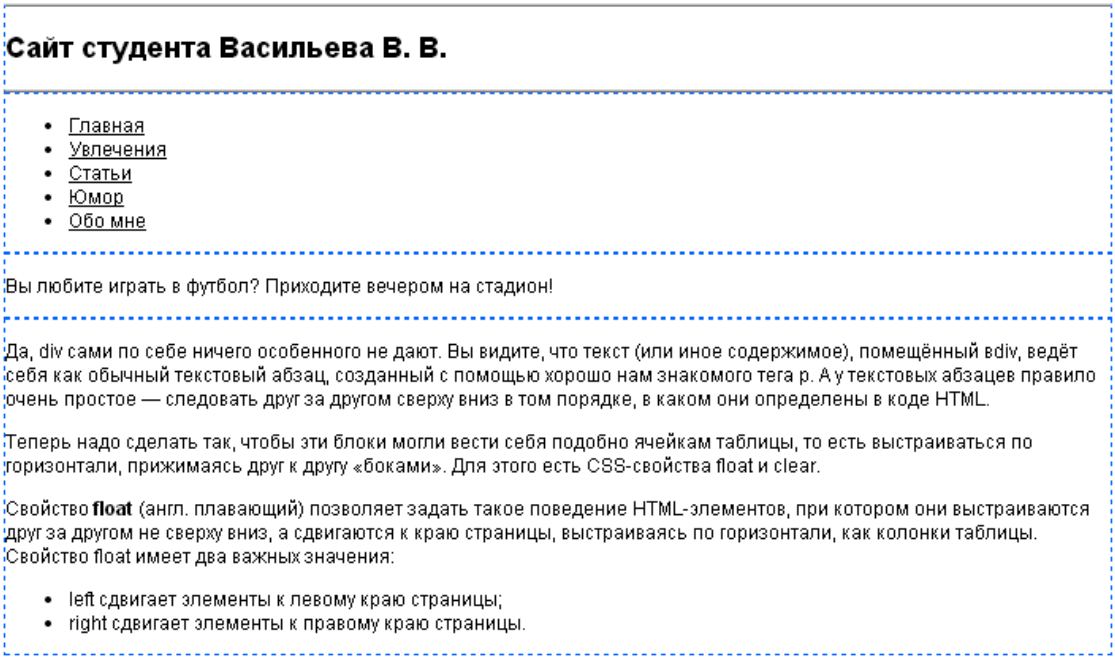


Рис.1 Размещение содержимого сайта в простых блоках `<div>`

Вы видите, что `<div>` сами по себе ничего особенного не дают. Текст, помещённый в `<div>`, ведёт себя как обычный текстовый абзац, созданный с помощью хорошо нам знакомого тега `<p>`. А у текстовых абзацев правило очень простое — следовать друг за другом сверху вниз в том порядке, в каком они определены в коде HTML.

Чтобы раскрыть всю мощь `<div>`, воспользуемся CSS-свойствами. Обычно блокам присваивают классы со следующими именами:

- `.header` («шапка», верхний колонтитул) для заголовка сайта;
- `.nav` (навигация) для навигационных ссылок;
- `.aside` (отступление, отклонение) для боковой панели;
- `.article` (статья) для основного содержимого.

С помощью CSS-свойства `width` ограничим навигационный блок шириной 120 пиксел, а блок новостей — 150 пиксел. Блоку со статьёй ширину не будем задавать, чтобы впоследствии вёрстка была «резиновой». Результат показан на рис. 2.

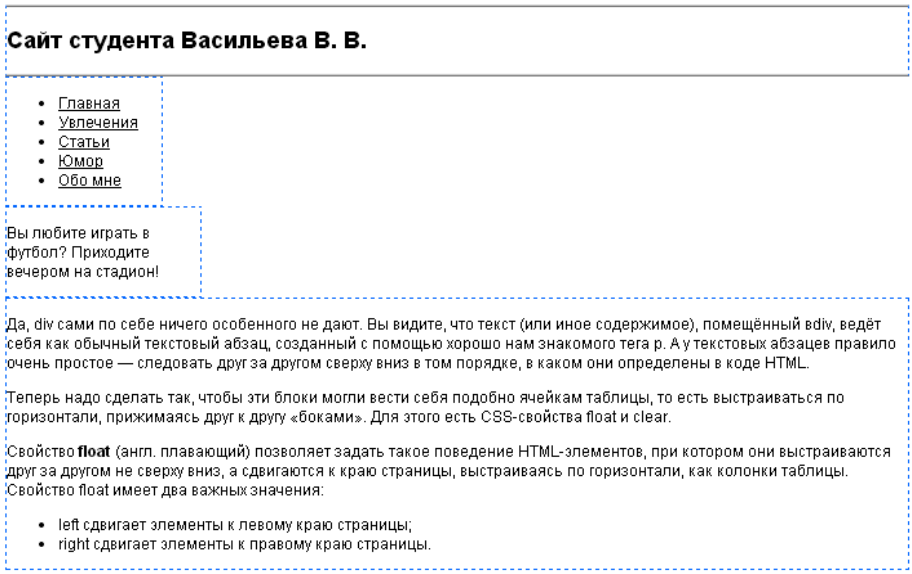


Рис. 2 Блокам `<div>` задали ширины

Теперь надо сделать так, чтобы блоки навигации и новостей прижались к левому и правому краям страницы, а статья расположилась между ними. Для этого воспользуемся CSS-свойством `float`.

Свойство `float` задаёт такое поведение HTML-элементов, при котором они сдвигаются к краю страницы и выстраиваются по горизонтали. Зададим свойство `float` со значением `left` блоку навигационного меню, а со значением `right` — новостям. Результат показан на рис. 3: навигационное меню осталось у левого края страницы (то есть «плавает влево»), блок новостей плавает вправо, а статья обтекает их.

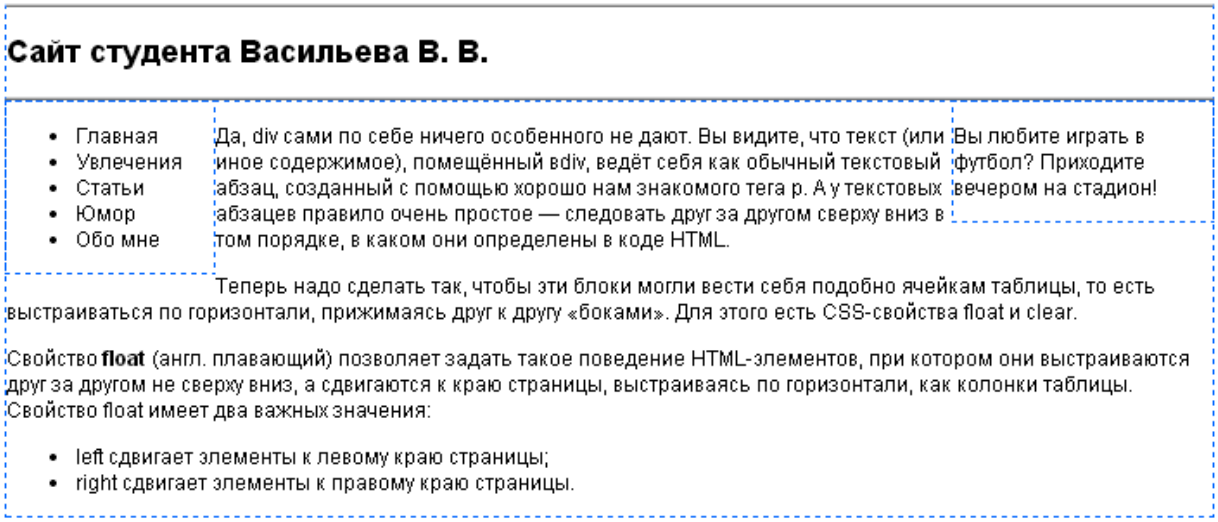


Рис. 3 Теперь блоки «плавают» влево и вправо

Три колонки уже обозначились, но выглядят они неряшливо. Продолжим левую и правую колонки вниз до самого конца страницы. Для этого с помощью свойства `margin` зададим статье поля слева и справа равные, или чуть больше, ширинам соответствующих колонок. Поле слева 130 пиксел, а справа 160 пиксел (рис. 4).

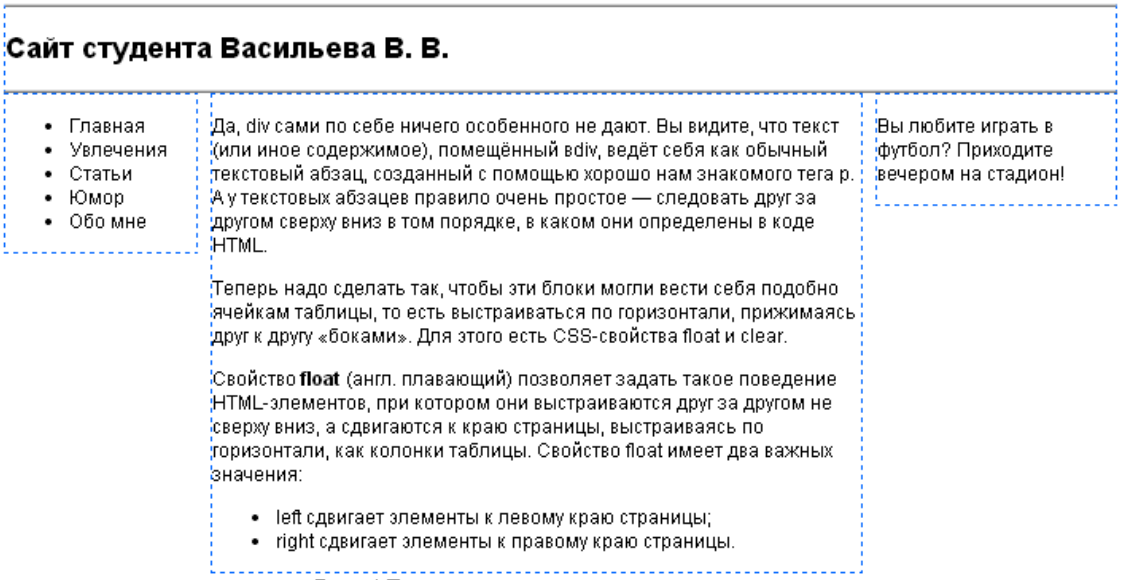


Рис. 4 Блок статьи имеет поля слева и справа

Вот теперь сайт смотрится аккуратно. Добавим блок, содержащий сведения об авторских правах, который должен находиться в самом низу страницы (в «подвале»). Для этого в коде страницы после блока со статьёй добавим ещё`<div>` с соответствующим содержимым (рис. 5). Обычно такому блоку присваивают класс с именем `.footer`(нижний колонтитул).

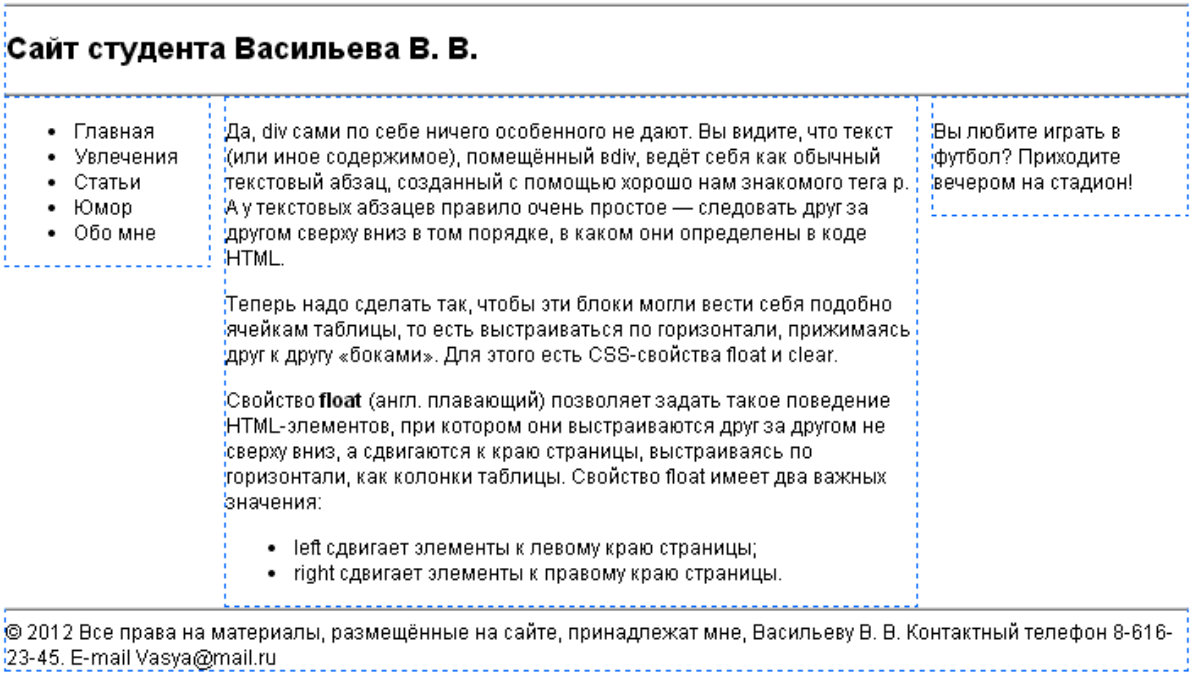


Рис. 5 Добавлен «подвал» с авторскими правами

Кажется всё уже хорошо и можно закончить. Но здесь есть один «подводный камень». Наш блок новостей «плавает вправо». Если понадобится добавить в него содержимое, и он станет больше статьи, то блок авторских прав начнёт его обтекать (рис. 6).

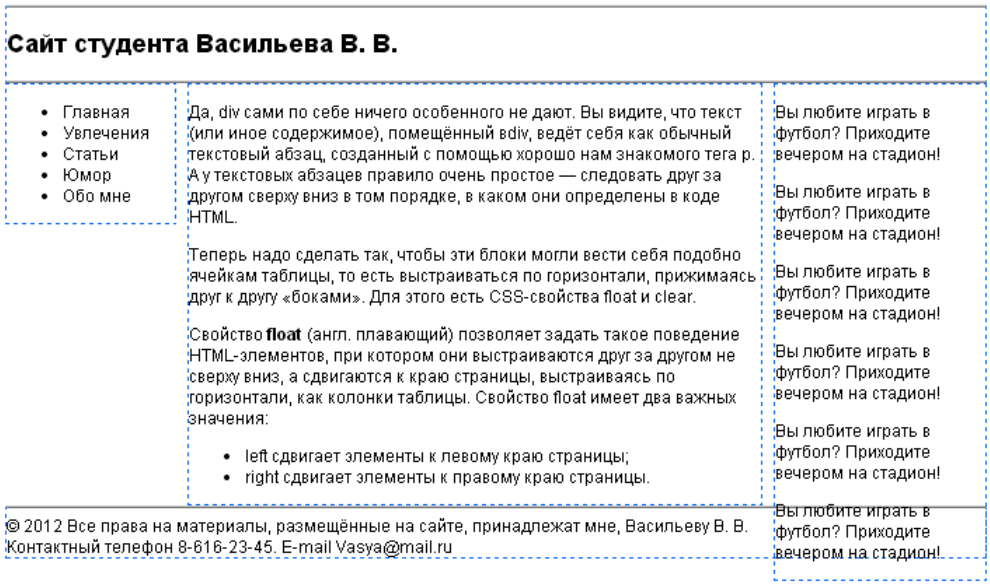


Рис. 6 «Подвал» обтекает блок новостей

Как же нам опустить «подвал» ниже «плавающих» блоков? Для этого воспользуемся CSS-свойством `clear` (англ. установленный в исходное состояние). Он позволяет сбросить обтекание предыдущих блоков как справа, так и слева. При этом блок располагается ниже остальных. Для задания такого поведения используется значение `both` — блок будет находиться ниже плавающих блоков, свойство `float` которых имеет значение `left` или `right`.

Вот теперь мы знаем, что делать. Нужно перед «подвалом» вставить новый пустой `<div>` и задать для него свойство `clear` со значением `both`. Такому блоку можно назначить класс с именем `.separator` (разделитель). После этого он разместится ниже трёх плавающих блоков (рис. 7).

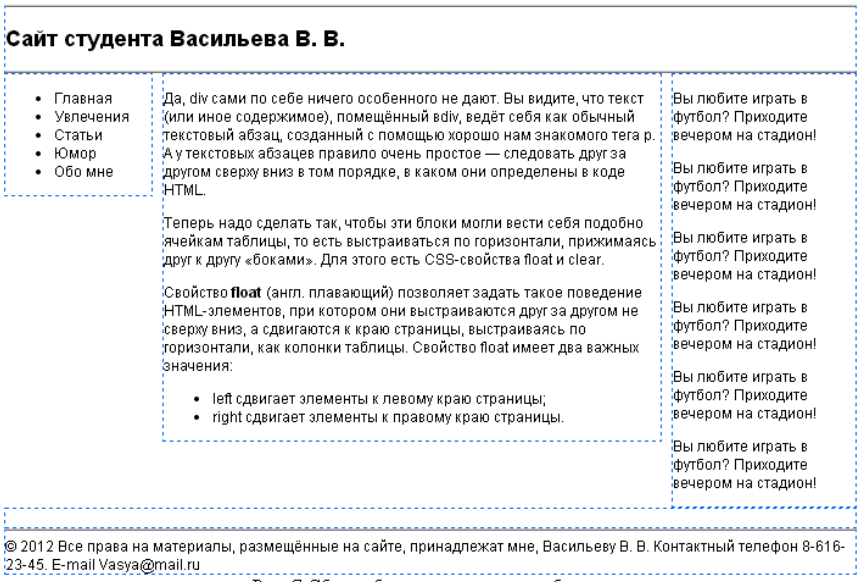


Рис. 7 Сброс обтекания плавающих блоков

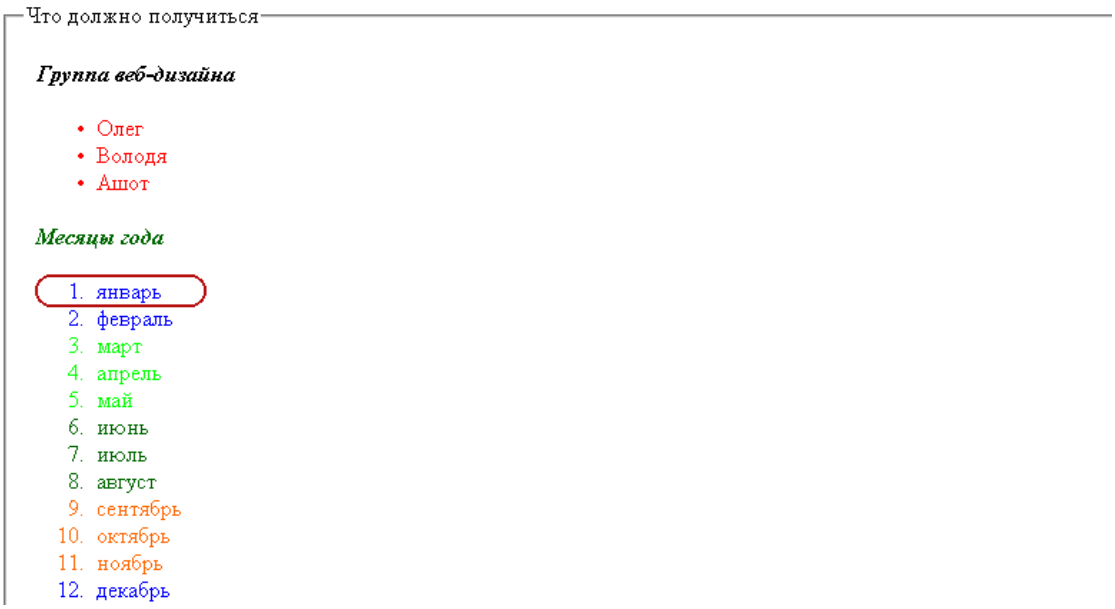
Таким образом, мы сверстали трёхколоночный сайт с помощью блоков `<div>`. Модульная сетка резиновая, то есть при увеличении или уменьшении размера экрана сайт заполняет всю ширину за счёт изменения ширины статьи.

Указания по выполнению

Задание 1

- Создайте страницу по образцу. На своей странице вставьте картинку с овалом (...\CSS\Материалы\ellipse.gif), задайте ей абсолютное позиционирование.
- Попробуйте в режиме дизайна разместить картинку над любым элементом, на свободном месте, сдвиньте за край страницы.
- Выделите овалом месяц январь как показано ниже.
- Добавьте ещё одно имя в списке группы веб-дизайна. Где оказалось выделение?

Что должно получиться

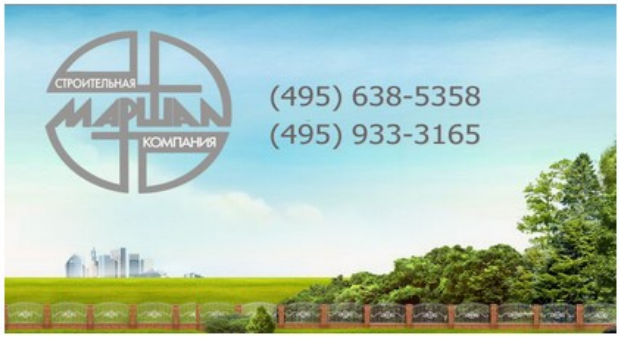


Задание 2

В сайте строительной компании разместите изображение города (файл city.png) и логотип (файл logo.png) поверх «шапки» сайта.

Что должно получиться

Что должно получиться



Задание 4

Создайте новую страницу orchestra.html и сверстайте страницу, показанную ниже. Текст и фотография находится по адресу ...\CSS\Материалы\Оркестр\.

Что должно получиться

Что должно получиться

Натан Рахлин

Про Рахлина справедливо говорили: «Богом данный талант». Он владел почти всеми инструментами оркестра. Автору этих строк посчастливилось присутствовать на репетеции «Болеро» Равеля, когда Натан Григорьевич, взяв тромбон у музыканта, показывал фразировку и ритмические акценты его соло. Звучала настоящая Испания — гордая и прекрасная. Это было грандиозно! А как виртуозно и музыкально играл Рахлин на гитаре! Он был ярче и убедительнее любого профессионала, и можно было слушать его музицирование бесконечно.

Невысокого роста, полноватый, Рахлин мало походил на стереотип дирижёра. Но он преображался, когда выходил на сцену. И от глубины и нестандартности его выдающихся трактовок, отточенности его жестов, совершеннейшей мануальной техники невозможно было оторваться.

«Особенно близка и дорога нам в дни войны замечательная музыка Глинки, Чайковского, Бородина, Мусоргского и других русских композиторов. Эта музыка ярко свидетельствует о духовном богатстве и моральном величии народов нашей страны, давших миру гениальные произведения»
(Н. Рахлин. «Вечерняя Москва»)

После того как Рахлин покинул пост главного дирижёра и художественного руководителя Госоркестра, он часто встречался с коллективом и в Москве, и на гастролях. И как всегда, с Рахлиным рождались незабываемые концерты, концерты-праздники.

В 1966 году Натан Григорьевич создал в Казани Государственный симфонический оркестр Татарстана, который уже через два-три года занял достойнейшее место среди симфонических коллективов тогда ещё большой нашей страны и который он возглавлял тринадцать лет, до самого ухода из жизни.

Жанна Дозорцева

Задания по вёрстке веб-страниц

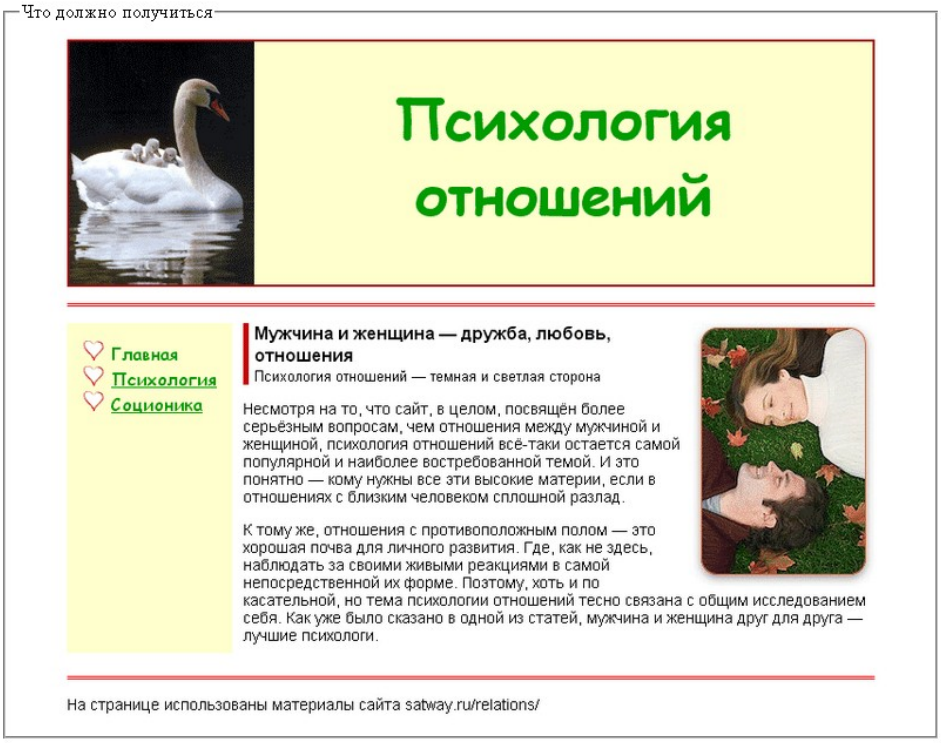
Задание 1

В новом файле psychology.html сверстайте страницу, показанную ниже, не используя ни одной таблицы. Картинки и текст находятся в папке ...\CSS\Материалы\Отношения\

- вёрстка резиновая;
- при наведении курсора на ссылки подчёркивание должно пропадать;
- шрифт названия страницы и навигационных ссылок — Comic Sans MS, остальной текст шрифтом Arial.

Что должно получиться

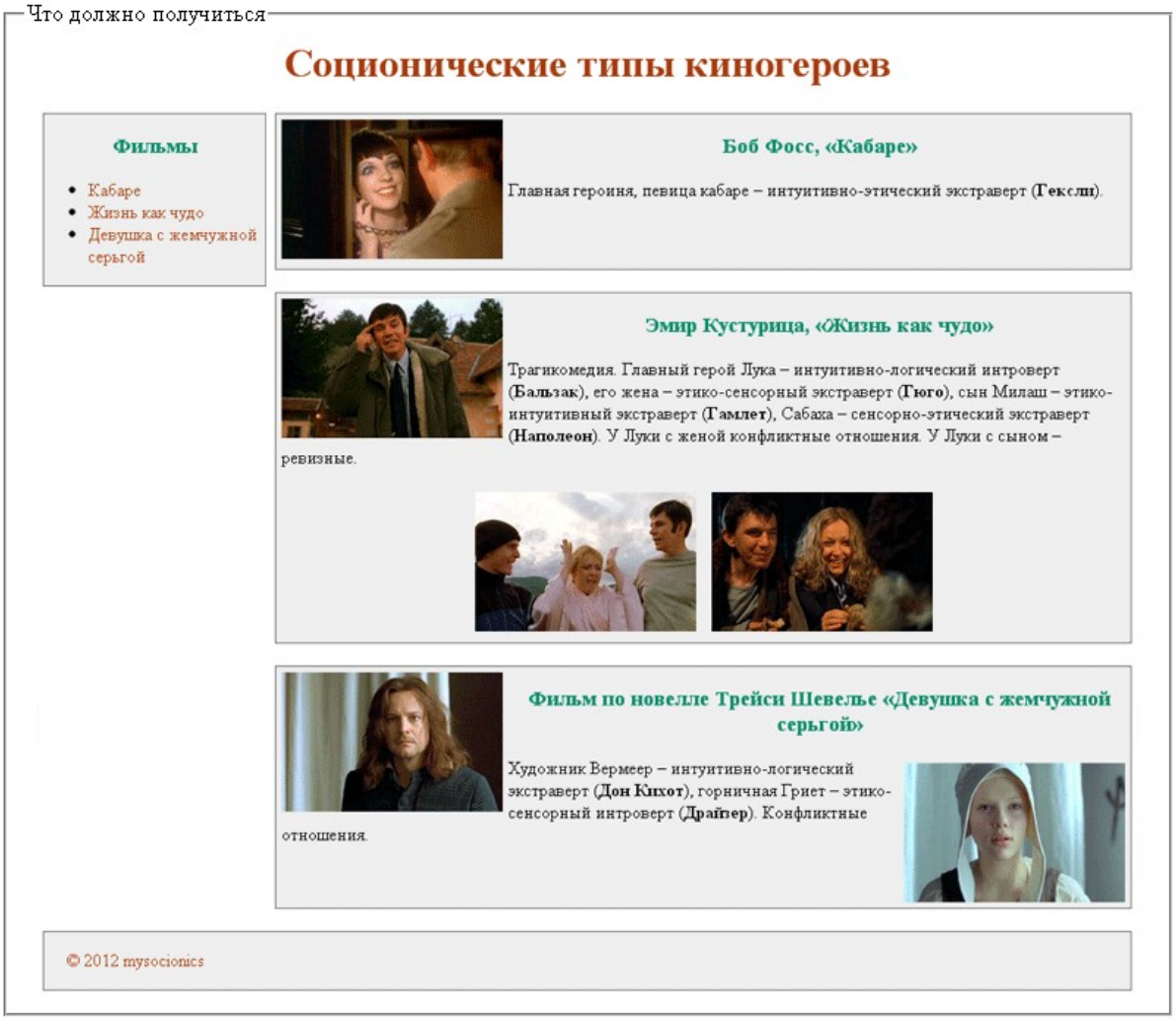
Что должно получиться



Задание 2

В новом файле films.html сверстайте страницу, показанную ниже, с помощью блоков. Картинки и текст находятся в файле ...\CSS\Материалы\Фильмы

Что должно получиться



Содержание отчета

19. Цель
20. Классы, id-стили
21. Позиционирование элемента, свойства CSS
22. Плавающий элемент: свойство, значения
23. Блочная вёрстка с помощью тега
24. Пример использования стилей
25. Выводы

Контрольные вопросы

9. Способы внедрения
10. Синтаксис CSS
11. Отступы (margin)
12. Свойства для позиционирования, задания плавающих элементов
13. Назначения тега
14. Какие имена классов обычно присваиваются блокам

Лабораторная работа №№ 12-13-14
Вёрстка страниц web-сайта

Цель: *Получить практические навыки проектирования и тестирования (исследования) статического веб-сайта с использованием текстового редактора.*

Ход работы:

13. Изучить теоретический материал.
14. Выполнить задания в соответствии с указаниями.
15. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
16. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Процесс проектирования сайта разбивается на следующие стадии:

1. Концептуальное проектирование
2. Логическое проектирование.
3. Физическое проектирование.

Этапы следуют последовательно один за другим, но в некоторых случаях возможен переход к следующей стадии без окончания предыдущей. Это может происходить, например, когда разработчиков несколько и каждый работает со своей частью сайта. В любом случае, после окончания этапа физического проектирования следует вернуться к началу и внести соответствующие коррективы.

Этапы проектирования

Концептуальное проектирование

Порой бывает сложно оценить эффективность сайта. Есть однако универсальный критерий, который довольно точно характеризует эффективность сайта. Это достижение разработчиками сайта поставленных перед ними целей. В этом случае сайт превращается в действенный инструмент, который выполняет возложенные на него функции.

Концептуальное проектирование служит для указания целей, задач сайта и определения аудитории, на которую он рассчитан.

На этом этапе следует описать следующее:

1. Основные и второстепенные цели.
2. Действия, которые необходимо предпринять для достижения поставленных целей.
3. Состав пользователей.
4. Интересы групп пользователей.
5. Разделы сайта.
6. Критерии достижения цели.

С учетом поставленных целей, а также интересов пользователей, в итоге получаем список сервисов и разделов, которые будут располагаться на сайте.

Логическое проектирование

Разделы сайта, продуманные на предыдущем этапе, пока не упорядочены и не структурированы, поэтому их нужно привести к удобному и понятному виду. Логическое проектирование включает организацию информации на сайте, построение его структуры и навигации по разделам.

На данном этапе следует задаться вопросом, каким образом будет упорядочена информация. Варианты могут быть самыми разными и зависеть от типа данных и

предпочтений создателей сайта: по времени, разделам, в алфавитном порядке, определенным группам или другим критериям. Так, для сайта музыкальной группы, поиск определенной песни можно сделать в виде алфавитного указателя, по названию альбома, первым строчкам песни, году выпуска и по ключевым словам. Одновременное использование различных способов охватывает большую аудиторию и позволяет быстрее найти нужную информацию на сайте.

На этом этапе следует описать следующее:

1. Тип структуры сайта (линейная, иерархическая, контекстная, другая).
2. Названия разделов.
3. Что будет содержать в себе каждый раздел.
4. Организация и связь разделов между собой.
5. Какая информация будет размещена на определенных страницах сайта.

Конечный результат логического проектирования оформляется в виде блок-схем, структурных диаграмм или другими способами, показывающими взаимосвязь различных частей сайта.

Физическое проектирование

Этап поиска проблем, а не их решений, связанных, по большей части, с технической реализацией сайта.

На этом этапе следует описать следующее:

1. Технологии, которые будут применяться на сайте.
2. Используемое программное обеспечение.
3. Возможные проблемы и способы их устранения.
4. Как будет обновляться информация.

После завершения данного этапа следует вернуться к концептуальному проектированию и проверить, не нужно ли внести изменения, в связи с переосмыслением проекта на других стадиях.

Так, если на сайте планируется использовать базы данных и доступ к данным с помощью CGI, следует подумать о хостинге, который поддерживает выбранные технологии. Возвращаясь к концептуальному проектированию, мы либо ставим себе цель разместить сайт на платном хостинге, затратив на это определенную сумму денег, при этом получая взамен дополнительные возможности. Либо окунуться в мир бесплатного хостинга. При этом часть возможностей просто теряется

Разработка – самый продолжительный и ответственный этап работы над интернет-ресурсом. Он включает в себя программирование, дизайн, подключение системы управления, наполнение материалами. Разработка дизайна обычно предполагает разработку нескольких вариантов, зачастую основанных на корпоративном стиле компании. В ходе работы должен проводиться анализ цветовых и ассоциативных предпочтений целевой аудитории. **Дизайн не должен увеличивать время загрузки страниц и замедлять работу интерактивных функций.**

Разработка структуры сайта необходима для:

- Создания четкой и логичной схемы навигации;
- Организации простой технологии внесения изменений при редактировании сайта.

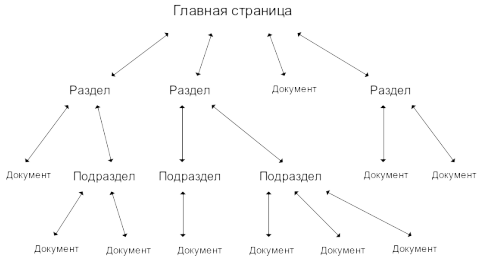
Для достижения этих целей процесс создания структуры принято рассматривать в двух аспектах. Фактически проектируются две структуры: логическая и физическая. Логическая структура определяет, в какой последовательности материалы будут доступны пользователю, какие ссылки следует выбирать для доступа к информации, размещенной на сайте. Хорошо продуманная логическая структура гарантирует, что на поиск необходимых данных будет затрачено меньше времени, и что они всегда будут найдены. Для создания полноценной логической структуры достаточно следовать нескольким простым правилам:

- Любой документ сайта должен оказываться доступным не более чем с помощью

трех переходов с главной страницы сайта;

- Все навигационные элементы должны отображаться сразу после загрузки страницы;
- Все внутренние связи должны быть двунаправленными, то есть позволять перемещаться между документами в обоих направлениях;
- С любой страницы должен быть предусмотрен возврат на главную страницу сайта;
- Названия рубрик и распределение материала между ними должно быть понятным каждому посетителю сайта.
- Если сайт имеет более одного уровня навигации, он обязательно должен содержать навигационную карту.
- Для удобства посетителей, каждый сайт должен иметь простую, четкую и логичную схему навигации.

Пример логической структуры веб-сайта показан на рисунке:



Логическая структура сайта

Существует несколько видов логических структур сайта. Самая простая из них – **линейная**. В ней страницы следуют одна за другой, и пользователь должен просматривать их последовательно, как слайд-шоу. Недостатков у такой структуры достаточно много. Как следствие, область ее применения ограничена. Она может использоваться на сайтах-презентациях и в онлайновых учебных пособиях. Самый простой вариант сайта с линейной структурой – набор страниц, с каждой из которых есть ссылка на следующую и на предыдущую. Однако и здесь начинающие авторы допускают ошибки. Дело в том, что на каждой странице обязательно должны присутствовать соответствующий заголовок и ссылка на первую страницу. В противном случае посетители, попавшие в середину сайта – например, в результате обращения поисковой системы, не смогут сориентироваться в ситуации, и покинут проект разочарованными. Кроме того, полезно показывать общее число страниц в отображаемом документе и выделять номер той из них, которая воспроизводится в данный момент времени.

Следующим вариантом структуры сайта является **линейная структура с альтернативами и вариантами**. Ее основой остается простое линейное размещение страниц. Однако на сайтах, построенных по данному принципу, посетители могут проявить некоторую инициативу, позволяющую облегчить поиск нужной информации. Под альтернативами здесь понимается выбор между двумя ветвями перемещения. Чаще всего подобная структура используется для сбора информации о посетителе. Примером может служить процесс регистрации клиента на сайте фирмы, оказывающей определенные услуги. Работа всегда начинается со стартовой страницы. Однако затем частным лицам предлагается ввести одну информацию, а представителям организаций – другую. Возможно, ветви альтернатив в дальнейшем вновь смыкаются, и все пользователи попадают на одну и ту же страницу.

Третий вариант носит название **линейной структуры с ответвлениями**. Это тоже контролируемая структура перемещения по ресурсам, наглядный аналог которой – дорога с ответвляющимися от нее тупиковыми тропами. Иными словами, посетитель последовательно переходит с одной страницы на другую. Если информация, размещенная на любой из них, его заинтересовала, и он желает ознакомиться с ней подробнее, то ему предоставляется возможность перейти на ответвление, а потом вернуться обратно на

основную "дорогу". Главное преимущество такой структуры состоит в том, что к ней легко перейти с обычного линейного размещения страниц. Часто бывает, что однажды созданный сайт перестает удовлетворять возросшим требованиям, а глобальная переделка по тем или иным причинам нежелательна. В этом случае автор может без затруднений расширить свой проект. И его сайт не утратит четкости логических связей.

Следующий вариант – **древовидная структура** – универсальный и в большинстве приложений наиболее предпочтительный способ размещения страниц. Она хорошо зарекомендовала себя для создания практически любых типов сайтов. Пользователь, попадая на главную страницу, оказывается перед выбором, в каком направлении двигаться дальше. После перехода в нужный раздел, он выбирает необходимый подраздел, затем пункт (параграф) и т.д. У древовидной структуры достаточно много достоинств, однако и она не лишена недостатков. Остановимся на главном из них. В древовидной структуре очень сложно соблюдать баланс между глубиной и шириной. Формальные критерии либо тривиальны, либо рассчитаны на узкие группы пользователей. Успех зависит от опыта, интуиции и квалификации автора. Если "дерево" сайта будет расти только вглубь, то посетителям, чтобы найти необходимую информацию, придется загрузить и просмотреть слишком много страниц. Естественно, это занимает много времени и раздражает. Если же создается очень широкая древовидная структура, то приходится тратить время по другой причине – для выбора нужной ветви поиска. Использование древовидной структуры сайта вынуждает постоянно следить за ее разрастанием и придерживаться золотой середины. Качество работы автора зависит от его мастерства, а не от соблюдения формальных правил.

Еще одним вариантом является **решетчатая структура**. Эта структура заметно сложнее рассмотренных ранее. В ней все страницы также размещаются в различных ветвях. Однако пользователю дается возможность перемещаться по ним не только вертикально (вверх и вниз), но и горизонтально (то есть между ветвями разных уровней). Используется «решетка» в основном только в каталогах. При этом перемещение между ветками на глубинных уровнях осуществляется с помощью ссылок на рубрики в других разделах.

Использование решетчатой структуры в других проектах считается нецелесообразным. Во-первых, она относительно сложна в реализации. Во-вторых, обращаться с «решеткой» нужно с очень большой осторожностью. В противном случае в схеме возникают непредусмотренные связи, и поиск информации приводит к непредсказуемым результатам.

Физическая структура не влияет на просмотр страниц посетителями и служит в основном для удобства создателя сайта при его редактировании, позволяя легко найти нужный файл (документ). При проектировании физической структуры разработчик может ненадолго забыть о пользователях и немного подумать об удобствах сопровождения сайта. Обычно распределяют отдельные папки для каждого из его разделов и подразделов. Внутри папок также создают отдельные папки для вспомогательных изображений, отдельные папки для мультимедиа-информации (музыка, видео, и т.п.) – то есть файлы сайта разделяются по функциональным признакам. В простом статическом сайте можно четко определить несколько групп файлов:

- Страницы сайта, представляющие собой html-файлы;
- Таблицы стилей;
- Клиентские скрипты;
- Графические файлы, используемые в дизайне сайта;
- Файлы для копирования посетителями.

Основные принципы создания сайта

Чтобы обратить на себя внимание посетителя, у сайта есть всего несколько секунд. Удержит ли пользователя главная страница Вашего сайта? В обществе насыщенном

информацией и многочисленными сайтами разной направленности главная страница сайта должна привлечь внимание посетителей и ответить на вопросы:

- Где?
- Что?
- Зачем?

Обычно основное внимание уделяют дизайну страницы. В браузерах мы видим только одну страницу за один раз. Сам сайт явно не представлен на экране.

Разработка архитектуры сайта сложная задача и имеет значение более важное, чем разработка дизайна страницы.

Обычно попадая на страницу, пользователь уже спустя короткое время понимает, что там можно сделать. Но перенести пользователя на нужную страницу не так просто.

Главная страница сайта

Главная страница это флагман сайта, и ее дизайн должен отличаться от дизайна всех остальных страниц. Конечно, первая и остальные страницы должны быть выдержаны в одном стиле, но есть и различия.

Например:

• На главной странице не должно быть кнопки “На главную”, так как не очень-то приятно нажимать на кнопку для того, чтобы попасть на ту же самую страницу.

• На главной странице обычно представлен более крупный логотип компании, ее название и название сайта.

Первая непосредственная цель любой главной страницы ответить на вопросы: “Где я нахожусь?” и “Что делает этот сайт?”

Оба они требуют прямого указания полного имени. Лучше, если и по дизайну будет понятно, какой цели может служить этот сайт для нового пользователя. Для нового пользователя, вероятно, самая важная функция первой страницы это ответ на вопрос “Что же этот сайт делает?”, а для большинства остальных первая страница отправная точка для навигации по сайту.

Первая страница это также место для представления новостей или специальных предложений, которыми вы хотите привлечь внимание всех посетителей.

Возможность поиска необходима на первых страницах, так как многие пользователи не хотят бесконечно ходить по ссылкам, чтобы попасть туда, куда им нужно.

Итак, главная страница должна предоставлять три следующие возможности:

• каталог основных содержательных разделов сайта (возможность навигации),

• краткую сводку основных новостей или специальных предложений,

• возможность поиска.

Если все это хорошо сделано, то каталог и новости помогут пользователю, пришедшему на сайт в первый раз, понять, о чем же, собственно говоря, этот сайт рассказывает. Всегда смотрите на главную страницу именно под таким углом: она должна отвечать на вопрос: “Что этот сайт может для меня сделать?”. И не забудьте о названии компании и логотипе.

Элементы web-страницы:

1. Заголовок/Логотип (Шапка)
2. Поиск
3. Рекламный Баннер
4. Контент /Содержание (Текстовое поле)
5. Элементы навигации
6. Информация о разработчиках сайта
7. Счетчик посещаемости

Любая web-страница содержит определенный набор стандартных элементов, являющихся обязательными компонентами каждого ресурса Интернета. Безусловно, ассортимент и количество подобных объектов могут варьироваться в зависимости от тематической направленности сайта, объема опубликованных на нем материалов, а также

от целей и задач, которые ставит перед собой создатель данного ресурса. Компоновка таких элементов, проектирование их взаимного расположения и составляет одну из главных задач web-мастера.

Первым элементом web-страницы, который нам предстоит рассмотреть, является ее **Заголовок**. Он может быть выполнен как в текстовом, так и в графическом варианте, однако и в том и в другом случае он должен располагаться в верхней части документа.

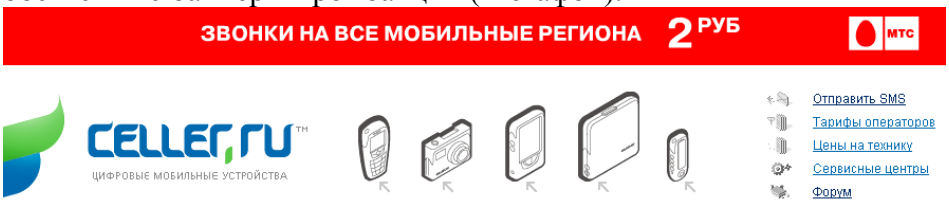
Логотип или название сайта выполняет для него ту же роль, что и вывеска на входе в какое-либо учреждение. Когда мы идем в магазин, достаточно на входе увидеть вывеску, чтобы внутри магазина точно знать, что я нахожусь в нем до тех пор, пока не выйду. Другое дело Интернет: здесь основной способ перемещения - это телепортация, и поэтому мне нужно видеть эту вывеску на каждой странице.

Мы ожидаем, что логотип сайта будет находиться в верхней части страницы, обычно в верхнем левом или правом.

Это объясняется тем, что логотип представляет весь сайт в целом, и, следовательно, он занимает самую верхнюю позицию в логической иерархии сайта. В визуальной иерархии каждой страницы можно сохранить эту доминирующую позицию двумя способами: либо сделать логотип самым заметным объектом на странице, либо поместить его так, чтобы он выполнял роль рамки для содержания страницы.

Иногда с заголовком совмещают меню выбора кодировки кириллицы и кнопки для перехода с русскоязычной на англоязычную версию сайта, если данный web-ресурс представлен на двух языках и **Навигацию**.

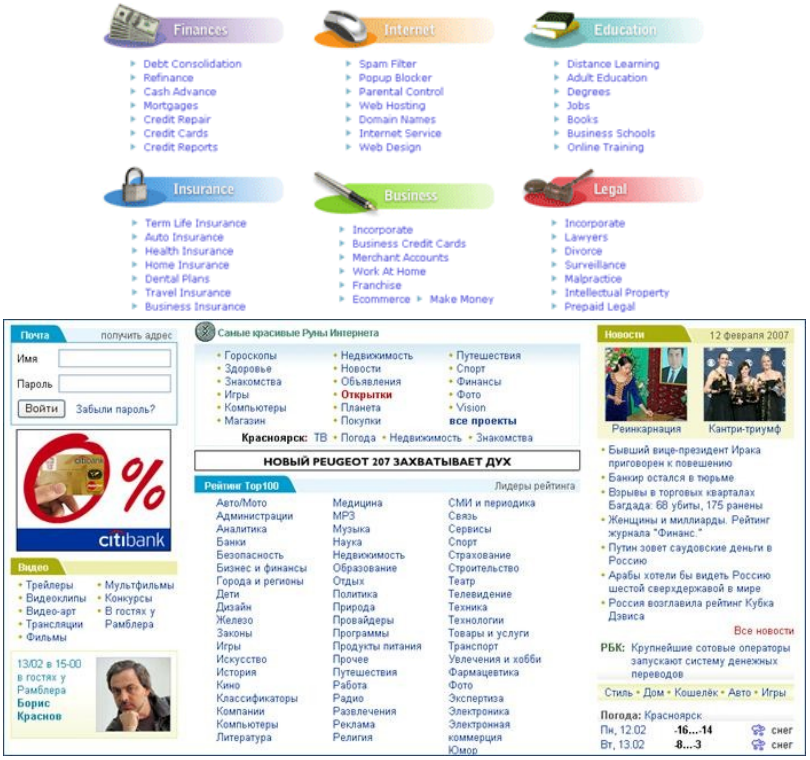
Непосредственно под/над заголовком документа, как правило, располагается пространство, отведенное для размещения рекламного **Баннера**. Включение баннера именно в верхнюю часть web-страницы в большинстве случаев является обязательным условием регистрации сайта в службах баннерного обмена - системах, рекламирующих созданный вами ресурс в обмен на показ на страницах вашего сайта рекламы других участников баннерообменной сети. В некоторых случая вместо заимствованного баннера включаются собственные баннеры промоакций (Мегафон).



Основную часть документа занимает так называемое **Текстовое поле** - участок, где и размещается смысловое наполнение страницы: содержательный информационный текст и иллюстрации.

Перечисленные элементы еще называют "Контент" (от англ, content - Содержание).

Расположение текстового поля зависит в первую очередь от того, каким образом web-дизайнер разместит остальные элементы документа.



Следующей обязательной составляющей частью web-страницы являются **Элементы навигации** - гиперссылки, связывающие данный документ с другими разделами сайта. Элементы навигации могут быть выполнены в виде текстовых строк, графических объектов, то есть кнопок, либо активных компонентов, например Java-апплетов. Последние представляют собой те же кнопки, которые умеют реагировать на движения мыши, выполняя при наведении на них курсора какие-либо несложные действия (включение подсветки, создание эффекта "нажатия", изменение формы и т. д.). Располагать элементы навигации следует таким образом, чтобы они всегда были "на виду", то есть так, чтобы пользователю не приходилось "отматывать" страницу назад, если текстовое поле занимает по высоте несколько физических экранов, после чего подолгу искать ссылки на другие разделы.

Наиболее устоявшимся подходом является размещение элементов навигации у левой границы страницы и/или верхней.

В нижней части документа принято публиковать **информацию о разработчиках сайта и адрес электронной почты**, по которому посетители ресурса могут направить владельцам странички свои отклики, предложения и пожелания.

Если web-страница является стартовым документом, в нижней ее части также размещают **счетчик посещений** - небольшой сценарий, вызывающий установленный на сервере CGI-скрипт, который фиксирует каждое открытие документа в браузере пользователей, изменяя значение индикатора счетчика. Благодаря этому web-мастер без труда определит количество посетителей, навестивших его страничку в течение какого-либо времени. Счетчик посещений устанавливается только на первой странице, вызываемой при обращении к сайту, в остальных документах ресурса он отсутствует. Не рекомендуется также размещать на одной странице несколько разных счетчиков.

Итак, мы разобрали все основные компоненты web-страницы. Пример компоновки сайта, содержащего набор описанных выше составляющих, показан на рисунках:

Пример компоновки web-страницы с левым позиционированием элементов навигации

Пример компоновки web-страницы с правым позиционированием элементов навигации

На практике часто встречаются web-сайты, в дизайне которых элементы навигации позиционированы по правой границе экрана. В этом случае текстовое поле смещается налево, остальные компоненты документа располагаются, исходя из принципа максимальной эстетичности их сочетания. Пример такого исполнения сайта показан на рисунке справа. Как видно из рисунка, логотип в этом случае помещен на один уровень с заголовком документа, а рекламный баннер позиционирован относительно центра страницы. При таком подходе рекомендуется выдерживать графическое оформление заголовка, логотипа и поля для размещения рекламы в едином цветовом и художественном стиле - тогда несимметричность положения данных объектов будет не столь очевидна и не станет "резать глаз".

Пример компоновки web-страницы с верхним позиционированием элементов навигации

Пример "смешанной" компоновки web-страницы

Элементы навигации можно разместить не только вблизи правой и левой границ страницы, но и в верхней части документа. В этом случае все объекты страницы гармонично "вписываются" в заданную ширину невидимой таблицы, при этом подготовка самой таблицы значительно упрощается. Единственным недостатком подобного

Это лишь общие принципы, которые применяются при компоновке структуры сайта, окончательное решение всегда остается за web-мастером.

Примером дизайнерского решения, не попадающим ни в одну из указанных выше категорий, может служить так называемая смешанная компоновка, примерная схема которой приведена на на рисунке выше.

подхода является необходимость продублировать элементы навигации в нижней части документа, поскольку при вертикальной прокрутке страницы они исчезают за верхней границей экрана, и, чтобы добраться до них, пользователю придется "отматыв ть" экран назад, что, согласитесь, весьма неудобно.

Как видно из рисунка, в данном примере часть управляющих элементов встроена непосредственно в заголовок. Основной блок элементов навигации позиционирован относительно левой границы документа и вв рху. Текстовое поле разделено на две несимметричные колонки, причем в правой размещены краткие анонсы составляющих ресурс тематических рубрик, включая ссылки на эти разделы.

Очевидно, что вариантов "смешанной компоновки" web-страницы может быть великое множество: конкретные решения зависят от количества составляющих ресурс разделов, объема подготовленного для размещения на сайте текста и, наконец, от фантазии самого дизайнера. Важно лишь, чтобы внешний вид сайта не вызывал нареканий у посетителей.

Поэтому перед тем как приступить к созданию своего сайта очень ответственно подойдите к проблеме компоновки элементов страниц.

В общем случае веб-страница делится на контент страницы, навигацию и сопутствующую информацию. Эти элементы в свою очередь делятся на более мелкие элементы.

Логическая и физическая структура сайта

Каждый ресурс Интернета, от домашней странички до большого информационного портала, содержит несколько тематических рубрик, соединенных между собой гиперсвязями. Как правило, ссылки на все разделы сайта с краткими анонсами их содержимого приводится на первой, так называемой стартовой странице, которой присваивается имя index.htm (.html). Если тематические рубрики содержат собственные подразделы, каждая из них также имеет свою стартовую страницу, называющуюся index.htm.

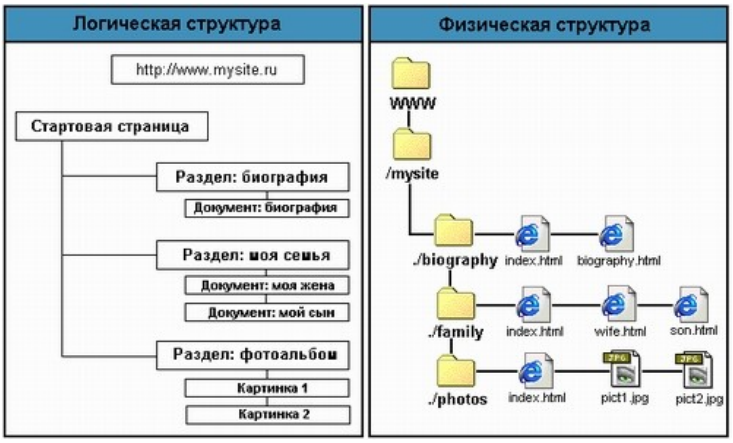
ПРИМЕЧАНИЕ!

Такое имя файла рекомендуется присваивать всем стартовым документам сайта, поскольку в противном случае при обращении к какому-либо разделу посредством сокращенного URL без указания названия стартовой страницы (например, http://www.mysite.ru/photos/ вместо http://www.mysite.ru/photos/startpage.html) браузер отобразит не саму web-страницу, а перечень хранящихся в данной папке файлов или ошибку.

Подобный набор тематических рубрик с распределенными по соответствующим разделам документами и заранее спроектированными гиперсвязями между всеми страницами ресурса и называется **Логической структурой сайта**.

Физическая структура подразумевает алгоритм размещения физических файлов по поддиректориям папки, в которой опубликован ваш сайт.

Пример сравнения логической и физической структур одного и того же ресурса Интернета показан на рисунке.



Сравнение логической и физической структуры сайта

Очевидно, что логическая и физическая структуры могут не совпадать, поскольку в общем случае физическая структура ресурса разрабатывается, исходя из удобства размещения файлов. Однако более или менее точное сохранение порядка следования логических разделов в физической структуре сайта позволит вам избежать путаницы при последующем дополнении и обновлении материалов.

СОВЕТ!

Рекомендуется размещать все графические изображения, являющиеся элементами проекта, в отдельной папке с названием "Images", “img” или “pic”, расположенной в корневой директории сайта. Такой подход позволит обновлять хранящиеся в других тематических разделах документы HTML без переноса графики, использовать одни и те же графические файлы во всех разделах сайта и при необходимости удалять целые директории.

Для того чтобы все гиперссылки на вашей домашней страничке или web-сайте работали корректно, все документы открывались правильно и браузер не выдавал ошибок при обращении к каким-либо разделам ресурса, при создании его физической структуры следует соблюдать несколько простых правил.

1. Назначайте имена директорий, имена и расширения документов HTML и графических файлов с использованием символов только латинского алфавита и только в строчном регистре.
2. Старайтесь, чтобы имена созданных вами файлов и директорий не превышали по длине восьми символов.
3. При присвоении имен файлов документам HTML старайтесь следить за тем, чтобы эти имена были "смысловыми": впоследствии вы легко можете забыть содержимое и назначение какой-либо web-страницы, если имена файлов будут выглядеть, например, как 1.htm, 2.htm, 3.htm и т. д.

Из всего сказанного становится очевидным, что физическая структура сайта скрыта от посетителей вашего ресурса: они могут наблюдать только логическую структуру, причем именно так, как она представлена при помощи элементов навигации. Отсюда следует вполне логический вывод: строение системы навигации должно если не полностью повторять, то хотя бы максимально соответствовать разработанной вами логической структуре сайта.

В итоге последовательность действий по разработке web-сайта сводится к следующему несложному алгоритму:

1. Постановка целей и определение основных задач.
2. Создание списка будущих тематических разделов.
3. Разработка логической и физической структуры ресурса.
4. Подготовка эскиза дизайна, компоновки сайта, невидимой верстальной таблицы.
5. Подготовка текстовых материалов.
6. Подготовка графических материалов в векторной форме.
7. Экспорт векторных рисунков в растровый формат.
8. Оптимизация всех изображений.

9. Создание шаблонов web-страниц.
10. Сборка web-страниц и отладка кода.
11. Проверка идентичности отображения web-страниц с различным экранным разрешением и цветовой палитрой и различных браузерах.

Названия страниц

Каждая страница должна иметь название в заголовке страницы

При подготовке информации для Интернета значительное внимание следует уделять вопросам, имеющим отношение к последующему поиску этой информации. Посетители любят пользоваться поисковыми системами как для поиска материалов в пределах текущего узла, так и для поиска в Интернете в целом. Для людей, использующих поисковые системы, тот или иной сайт существует исключительно в виде названия, отображаемого в результатах поиска.

HTML позволяет указывать в разделе заголовка страницы ее название. Названия web-страниц имеют большое значение, поскольку они нередко выступают в виде ссылок на соответствующие страницы. Кроме того, названия страниц отображаются во всевозможных меню навигации в броузерах, таких как списки закладок (избранные сайты) или ранее посещенные сайты.

Во многих из перечисленных ситуаций названия web-страниц рассматриваются в отрыве от основного контекста. Таким образом, необходимо, чтобы названия страниц были достаточно информативными и позволяли правильно судить о характере содержащихся на этих страницах материалов. С другой стороны, слишком длинные названия также неудобны.

Название страницы также можно отнести к разряду информационного наполнения web-пространства. Каждое название должно представлять собой образец четкости формулировки. Автор должен объяснить потенциальному посетителю, какую информацию можно найти на соответствующей странице.

Каждая из страниц должна иметь уникальное название. Представьте себе ситуацию, когда вы посетили, скажем, семь страниц с одинаковыми названиями, а затем пытаетесь вернуться к одной из них, используя список посещенных страниц. Кроме того, серьезные неудобства возникают при создании закладок на несколько различных страниц подобного узла, поскольку при этом меню закладок (избранных страниц) будет содержать несколько идентичных элементов.

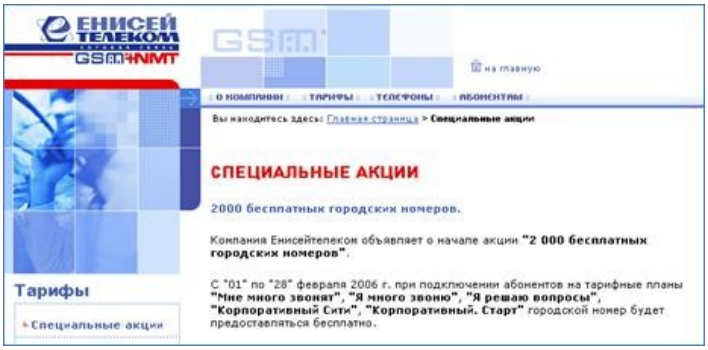
И наконец, названия должны быть приспособлены для беглого ознакомления. С этой целью следует помещать наиболее информативные фразы в начало названия, причем предпочтительнее всего начинать название со слова, которое с наибольшей вероятностью будет отражать потребности потенциального посетителя. Весьма часто встречается, когда в качестве названия используются такие фразы, как “Добро пожаловать на узел ZZZ”. Гораздо лучшим вариантом было бы использование в качестве названия просто “ZZZ”. Названия не обязательно должны представлять собой законченные грамматические структуры, скорее они должны быть выдержаны в духе рекламных слоганов.

При создании заголовков в Интернете необходимо руководствоваться приводимыми ниже принципами:

1. Заголовок должен четко описывать содержание соответствующей страницы.
2. Заголовок должен быть написан понятным языком.
3. Первое слово должно быть наиболее информативным, что будет способствовать оптимальному позиционированию в алфавитных списках и облегчит беглое ознакомление. Например, можно начинать заголовки с названия организации, имени автора или понятия, представляющего собой предмет обсуждения.
4. Названия страниц не должны начинаться с одного и того же слова. Такие названия будет сложно различать при беглом просмотре списка. Общие слова лучше всего помещать в конец названий.

Каждая страница должна иметь название и в дизайне

Название страницы должно находиться в правильном месте. Другими словами, название страницы должно занимать в ее визуальной иерархии такое место, чтобы оно обрамляло расположенное на ней содержание. (В конце концов, в этом состоит основная задача именно названия, а не навигации или рекламы, которые, в свою очередь, представляют всего лишь инфраструктуру сайта).



Название страницы должно быть заметным.

Позиция, размер, цвет и шрифт названия должны ясно показывать, что это заглавие для всей страницы в целом. Поэтому, в большинстве случаев, текст названия страницы будет иметь самый большой размер.

Название страницы должно соответствовать названию ссылки, по которой мы щелкнули.

Существует негласное правило о том, что название страницы соответствует тем словам, по которым пользователь щелкнул мышью для того, чтобы на эту страницу попасть.

Обратите внимание на пример страницы сотовой компании ЕТК. Вы без труда определите, в каком разделе сайта вы находитесь, название страницы.

Удобочитаемость

Если пользователь оказывается не в состоянии прочитать текст, все остальное — дизайн, быстрота загрузки, информационное наполнение — не имеет никакого смысла. Существует ряд основных правил, которых следует придерживаться для обеспечения разборчивости при публикации материалов.

1. Фон и текст страницы должны быть оформлены контрастными цветами. Наилучшая разборчивость обеспечивается при использовании черных символов на белом фоне (так называемого позитивного текста). Также хорошо воспринимается белый текст на черном фоне (негативный текст). Несмотря на то, что уровень контрастности в последнем случае аналогичен позитивному тексту, негативная цветовая схема способствует некоторому рассеянию внимания читателя и очень замедляет чтение. Разборчивость еще больше страдает, если цвет текста несколько светлее черного, в особенности, если при этом фоновый цвет несколько темнее белого. Наиболее **неудобны** для чтения такие цветовые схемы, как розовый цвет на зеленом фоне: эта схема характеризуется слишком низкой контрастностью и, кроме того, не позволяет читать текст пользователям, страдающим нарушением восприятия красного и зеленого цветов.

2. В качестве фона страницы следует использовать либо однотонный цвет, либо узор, имеющий минимальную контрастность. Фоновые изображения затрудняют распознавание текста глазом человека.

3. Текст должен быть набран достаточно крупным шрифтом, чтобы его могли читать даже пользователи с ослабленным зрением.

4. Текст должен быть статичным. Движущийся, мерцающий или изменяющийся в размерах текст воспринимается значительно хуже, нежели статичный.

5. В большинстве случаев текст должен быть выровнен по левому краю или по ширине. Если начало всех строк находится на одном уровне по горизонтали, чтение текста значительно ускоряется в сравнении с текстом, выровненным по центру или по правому краю. Конечно, выравнивание по правому краю или по центру отдельных небольших фрагментов текста вполне допустимо, однако основной текст должен быть выровнен по левому краю или по ширине.

6. Аналогично, восприятие списков значительно облегчается при выравнивании первого слова каждого из элементов списка по левому краю вдоль одной линии.

7. Поскольку современные мониторы имеют сравнительно низкую разрешающую способность, текст, набранный мелким шрифтом, воспринимается значительно лучше при оформлении его с использованием рубленых шрифтов, таких как Verdana. Для четкого отображения засечек шрифта размером 10 пунктов попросту не хватает пикселов. В то же время, большинство людей предпочитает читать текст, набранный шрифтами с засечками, таким образом, мы оказываемся перед лицом своего рода парадокса. Разборчивость должна быть определяющим критерием в случае очень мелких шрифтов (9 пунктов и менее). Такой текст должен набираться с использованием рубленого шрифта.

8. Не следует набирать текст ПРОПИСНЫМИ БУКВАМИ. Чтение такого текста выполняется примерно на 10% медленнее, чем текста, набранного с использованием строчных букв, поскольку восприятие символов, имеющих одинаковую высоту, затруднено. Этого следует избегать.

Разработка информационного наполнения

В конечном итоге, основной целью посещения любого сайта является получение определенной информации. Все остальное лишь дополнение, и смысл дизайна оказать людям необходимую помощь в получении доступа к этой информации.

Характер текста веб страницы определяет не только ее содержание, но впечатление, производимое на пользователя, поскольку посетители в первую очередь обращаюn внимание на текст и заголовки. Отсутствие в тексте грамматических ошибок имеет большое значение, однако также немаловажно представлять информацию в форме, позволяющей акцентировать внимание посетителя на наиболее существенных моментах повествования.

При создании текстов для Сети нужно придерживаться трех основных рекомендаций:

1. Следует соблюдать лаконичность. На странице: должно размещаться не более 50% текста, который может быть использован для передачи того же материала в печатном издании.

2. Текст должен быть удобным для беглого ознакомления. Не следует заставлять посетителя читать крупные абзацы текста. Вместо этого лучше использовать небольшие абзацы, подзаголовки и маркированные списки.

3. Информацию значительного объема следует разбивать на несколько страниц, связанных между собой гиперссылками.

Исследования показали, что чтение текста с экрана монитора происходит приблизительно на 25% медленнее, чем чтение печатного текста. Даже те пользователи, которые не знакомы с результатами подобных исследований, отмечают, что они испытывают некоторые неудобства при чтении информации, представленной в электронном виде. В результате этого люди избегают чтения больших объемов текста с экрана монитора. Таким образом, объем текста следует снижать на 50% (снижение объема на 25% оказывается недостаточным), поскольку в данном случае значение имеет не только скорость чтения, но и удобство восприятия информации. Также известно, что пользователи не любят прибегать к прокрутке, что является еще одной причиной для сокращения объема страниц.

Рисунки и фотографии

1. Количество графической информации на web-страницах необходимо максимально ограничивать, поскольку загрузка графики требует значительного времени. От излишней графики следует попросту отказаться. В разряд такой графики попадает любая текстовая информация, представленная в графическом формате, за исключением текста, тесно связанного с общей концепцией оформления сайта, включение которого в состав изображения обусловлено необходимостью.

2. Несмотря на это, пользователям нужно видеть фотографии образцов предлагаемой продукции, поскольку это является единственным способом познакомиться с ней воочию. На персональной странице целесообразно помещать фотографию или текст, говорящие о личности автора. Устранить противоречия между этими двумя требованиями помогают гипертекстовые возможности Сети. Количество графической информации на страницах верхних уровней следует сводить к минимуму. При просмотре этих страниц пользователь еще не успевает выбрать материалы, которые представляют для него интерес и нуждаются в иллюстрациях. На страницах, посвященных более узким вопросам, количество иллюстративного материала может быть увеличено.

3. На странице верхнего уровня, посвященной определенной продукции, можно поместить небольшую фотографию образца этой продукции, однако основной объем информации по-прежнему должен быть представлен текстом и табличными данными. Если пользователь действительно заинтересовался описываемым образцом, он может воспользоваться ссылками на странице описания для просмотра других фотографий этого образца. Эти фотографии должны быть достаточно крупными, чтобы давать потенциальному покупателю максимально полное представление о предлагаемой продукции.

Уменьшение размера изображения

Традиционным способом создания небольших версий изображений является уменьшение исходного изображения в графическом редакторе. К сожалению, при изменении масштаба изображения уменьшаются настолько, что становятся абсолютно неразборчивыми. Применение кадрирования (обрезки краев изображения) позволяет сохранить детализацию на исходном уровне, однако при этом утрачивается определенная часть содержащейся в изображении информации. Оптимальным выходом в данной ситуации может стать совместное использование обоих указанных методов. Например, для получения уменьшенного образца, размер которого составляет 10% от размера исходного изображения, вначале следует обрезать изображение до 32% относительно исходного текста, расположенного в центре страницы, если в верхнем углу помещен переливающийся логотип.

Анимированные изображения имеют право на существование в web-дизайне, однако в общем случае использование анимации лучше всего ограничивать. Задайте себе вопрос: можно ли достичь той же цели без применения анимации? Если ответ на этот вопрос окажется положительным, избавляйтесь от анимации без малейшего сожаления.

Кроме того, нельзя допускать воспроизведения анимации в бесконечном цикле достаточно, чтобы она воспроизводилась всего несколько раз. Отдельные пользователи полагают, что анимированные изображения выглядят эффектно и являются показателем труда, вложенного в разработку сайта. Тем не менее, большинство пользователей отмечает, что анимация раздражает их. В частности, практически все не переносят движущийся текст и бегущие строки.

Применение анимации имеет семь целей:

1. для передачи переходных процессов;
2. для указания направленности действия;
3. для передачи изменений, происходящих с течением времени;
4. для смены, отображаемой в отдельной области страницы информации;
5. для обогащения графического представления;
6. для визуализации объемных структур;
7. для привлечения внимания.

Описания ссылок

Ссылки самая важная часть гипертекста. Они объединяют страницы и позволяют пользователям путешествовать по Сети. Существуют три основные формы ссылок:

• Структурные ссылки или ссылки на элементы навигации. Эти ссылки задают структуру информационного пространства, и по ним пользователь переходит к другим

разделам сайта. Типичный пример кнопка возврата на начальную страницу и ссылки на подчиненные страницы.

• Ассоциативные ссылки внутри страницы. Чаще всего это подчеркнутые слова в тексте (хотя это могут быть и изображения), указывающие на страницы, где можно найти подробную информацию о слове, играющем роль ссылки.

• Списки типа “смотрите также”. Эти ссылки нужны для того чтобы помочь пользователю найти то, что он хочет, но чего нет на данной странице. Учитывая сложность навигации в Интернете, хорошо составленные списки “смотрите также” могут сильно облегчить жизнь пользователю.

Гипертекстовые ссылки привязаны к тексту, по которому щелкает пользователь, чтобы перейти по ссылке. Этот текст не должен быть очень длинным, так как пользователи проглядывают страницы в поисках ссылок, стремясь узнать, что они могут сделать на данной странице. Ссылки выполняют ту же роль, что и выноски в печатных журналах: именно на них останавливается взгляд пользователя при просмотре страницы.

Если слишком много слов играют роль ссылок, то пользователь не сможет понять их смысл при беглом взгляде на страницу. Гиперссылками должны быть только понятия, несущие важную информацию.

Подчеркивать информационно значимые слова важно. Но еще лучше добавить краткий текст, поясняющий тип предлагаемой дополнительной информации.

Хотя сама гиперссылка не должна быть длиннее четырех слов, нелишним будет добавить описание, поясняющее смысл ссылки. Особенно это касается ссылок, которые выглядят почти одинаково. Чтобы пользователь мог понять, по какой именно ссылке он может найти нужную ему информацию, нужно дать короткое описание каждой из таких ссылок.

Заголовки ссылок

Начиная с броузера Internet Explorer 4.0 поддерживается возможность выводить всплывающие подсказки с описанием ссылки. Это помогает узнать, куда приведет ссылка, и улучшает возможности навигации.

Описание ссылки называется *заголовкомссылки*и его очень легко закодировать.

Если вы поместите указатель мыши в вашем браузере над этой ссылкой, то примерно через секунду появится всплывающая подсказка.

Цель заголовков помочь пользователям предугадать, что произойдет, если они перейдут по ссылке. В качестве заголовка для ссылки можно включать следующую информацию:

• Название сайта, на который ведет ссылка (если это не текущий сайт).

• Название подраздела сайта, на который ведет ссылка (если она не выводит за пределы сайта).

• Дополнительные сведения об информации той страницы, на которую указывает ссылка, или о том, как она связана с текстом ссылки и содержимым текущей страницы.

Заголовки ссылок должны быть меньше 80 символов, но они редко превышают даже 60 символов. Чем короче заголовок, тем лучше.

Кроме того, вы не должны придумывать заголовки для всех ссылок. Если из текста ссылки и контекста понятно, куда она ведет, то ее заголовок только уменьшит удобство использования сайта, так как это будет лишний элемент, на который должен будет посмотреть пользователь. Излишне просто повторять текст ссылки в заголовке.

Не думайте, что заголовок ссылки будет выглядеть одинаково для всех пользователей. На самом деле браузеры с голосовым интерфейсом прочитают заголовок ссылки вслух, а не отобразят ее на экране.

И наконец, имейте в виду, что использование заголовков ссылок не заменяет сами ссылки. Текст ссылки и контекст, в котором она приводится, должны быть понятны, даже если пользователь и не посмотрит на ее заголовок.

Сделайте ссылки цветными

Большинство браузеров используют два цвета для отображения ссылок: ссылки на страницы, которые пользователь не посещал раньше, обычно отображаются синим цветом, ссылки на страницы, где пользователь уже бывал, отображаются красным или пурпурным цветом. Для удобства пользования сайтом необходимо всегда оставлять эти цвета. И хотя не обязательно использовать те же оттенки, что заданы в браузере по умолчанию, необходимо, чтобы не посещенные ссылки были бы синими, а посещенные красными.

Синий цвет сложнее читать, чем черный или красный (предполагаем, что фон белого цвета), так как в человеческом глазе меньше рецепторов, воспринимающих длину волны, соответствующую синему цвету. Но, несмотря на этот физиологический факт, рекомендуют использовать для обозначения ссылок синий цвет. Причина проста: люди привыкли к тому, что синий цвет это цвет ссыпки, поэтому они без заминок разберутся, как работать со страницей.

Когда используются нестандартные цвета для ссылок, люди не могут точно определить, какие части сайта они уже посетили, а какие еще нет. Чувство структуры и положения на сайте у пользователя сильно ослабевает, и в результате страдает его способность обдуманной навигации.

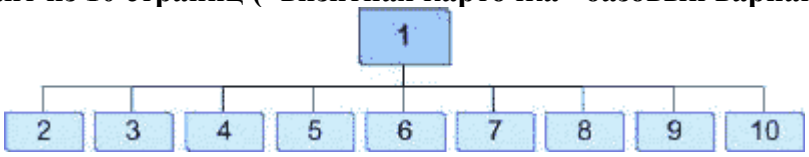
Ожидание от перехода по ссылке

Всегда используйте один и тот же URL, когда ссылаетесь на какую-то страницу. Если в одной ссылке используется один URL, а в другой ссылке другой, браузер не сможет узнать, что обе ссылки ведут на одну и ту же страницу. Поэтому, даже если пользователь перейдет по одной из них, другая будет отображаться как непосещенная — разумеется, это собьет человека с толку, так как он на самом деле уже посещал эту страницу.

Структура сайта

Ниже приведена структура сайта различного. В каждом конкретном случае структура сайта, название и взаимное расположение отдельных страниц может отличаться в зависимости от задач, решаемых сайтов, его объёма и содержания

Сайт из 10 страниц ("визитная карточка" базовый вариант)



Такой сайт может включать в себя до 25 страниц текста формата А-4 и до 50 фотографий (рисунков), выполненных с предпросмотром.

Здесь:

- 1 - Главная (домашняя) страница;
- 2 - Прайс-лист;
- 3 - Фото (каталог) товаров;
- 4 - Справочная информация;
- 5 - О фирме;
- 6 - Офис;
- 7 - Партнёры;
- 8 - Вакансии;
- 9 - Потребности;
- 10 - Сервисы.

Примеры форматирования страницы с помощью CSS.

Двухколоночный макет

HTML код:

```
<div id="container">
<div id="content">Это - некоторое <br />содержимое</div>
<div id="rail">Это – правая колонка</div>
</div
```

Код CSS:

```
#container{
background-color:#FFFF00;
overflow:hidden;
width:750px;
}
#content{
background-color:#FFFF00;
margin-right:-150px;
float:left;
/* Ширина и цвет правой колонки */
width:600px;
border-right-width: 150px;
border-right-style: solid;
border-right-color: #FF00FF;
}
#rail {
background-color:#FF00FF;
width:150px;
float:left;
}
```

Правая граница в div content такой же ширины и цвета, как и блок rail, блок rail смещён свойством float влево. Правая граница margin-right:-150px блока content позволила блоку rail сместиться влево на освобожденное место. Если блок content будет длиннее блока rail, вместе с ним вырастет и граница, блок rail тоже вытянется. Фон блока container и content одинаковый для того, если блок rail является более высоким, то блок container *“вырастет”* вместе с ним. Для того, чтобы блок rail выровнялся по левому краю необходимо внести небольшие изменения только в таблицу стилей CSS в зависимости от того, что длиннее, либо content, либо rail.

Трёхколоночный макет

Трехколоночный макет требует немного другого подхода, а именно граница применяется непосредственно к блоку container.

HTML код:

```
<div id="container">
<div id="center">Это - некоторое <br />содержимое</div>
<div id="leftRail">Левая колонка</div>
<div id="rightRail">Правая колонка</div>
</div>
```

Код CSS:

```
#container{
  background-color:#0ff;
  float:left;
  width:500px;
  border-left:150px solid #0f0;
  /* Ширина и цвет левого блока */
  border-right:200px solid #f00;
  /* Ширина и цвет правого блока */
}
#leftRail{
  float:left;
  width:150px;
  margin-left:-150px;
  position:relative;
}
#center{
  float:left;
  width:500px;
  margin-right:-500px;
}
#rightRail{
  float:right;
  width:200px;
  margin-right:-200px;
  position:relative;
}
```

Центральная колонка имеет границу -500px. Это позволяет левому блоку rail располагаться с левого края центральной колонки. Отрицательные границы вставляют боковые колонки на свои места. Существует несколько способов добиться этого, но данный кажется наиболее оптимальным решением при переходе позже к *«тянущемуся»* макету.

Для блока container установлено свойство float: left вместо overflow:hidden. Сделано это потому, что колоночные блоки расположены за пределами блока container, обтекая его границы, поэтому они просто исчезнут с использование свойства overflow: в IE боковые колонки не исчезают, однако в Firefox корректно пропадают.

Для колонок не нужно устанавливать цвет фона. Поскольку цвета устанавливаются для блока container, который «растет» вместе с самой длиной колонкой, появляется иллюзия равной высоты всех колонок.

«Тянущийся» *макет*

После того, как мы разобрались с фиксированным макетом, перейдём к созданию *«тянущегося»* макета, используя тот же самый метод. Боковые колонки должны по-прежнему иметь фиксированную ширину, поскольку большинство браузеров не поймет просто процентное задание ширины границ, но мы можем создать *«резиновую»* центральную колонку.

HTML код:

```
<div id="container">
<div id="content">Это - некоторое <br />содержимое</div>
<div id="rail">Это - правая колонка</div>
</div>
```

Код CSS:

```
#container{
        background-color:#0ff;
        overflow:hidden;
        padding-right: 150px;
}
* html #container{
        height:1%; /* Для IE */
}
#content{
        background-color:#0ff;
        width:100%;
        border-right:150px solid #f00;
        float:left;
        margin-right: -150px;
}
#rail{
        background-color:#f00;
        width:150px;
        float:left;
        margin-right: -150px;
}
```

Размета точно такая же, как и при двухколоночном макете фиксированной ширины, за исключением блока content. Таблица стилей CSS претерпела незначительные изменения, однако обратим внимание на несколько важных различий: блок container не имеет фиксированной ширины и добавлена высота в 1%, чтобы IE правильно обработал свойством overflow:hidden.

Отступ справа padding-right создает дополнительное пространство для блока и блок content заполняет все что осталось, поскольку имеет ширину 100%, если бы не было отступа, то боковой блок ушел бы за пределы блока container и исчез бы. Наконец, ширина блока content переключается на 100% и добавляется граница к боковой колонке margin-right:-150px.

Трехколоночный *«тянущийся»* макет

HTML:

```
<div id="container">
  <div id="center">Центральное содержимое</div>
  <div id="leftRail">Левая<br />колонка</div>
  <div id="rightRail">Правая колонка</div>
</div>
```

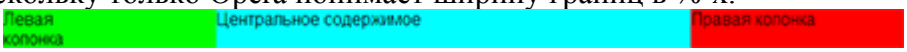
The CSS:

```
body{
        padding-top: 0;
        padding-bottom: 0;
        padding-left: 200px;
        padding-right: 200px;
}
#container{
        background-color:#0ff;
        float:left;
        width:100%;
        border-left:200px solid #0f0;
        border-right:200px solid #f00;
        display:inline; /* Для IE */
        margin-right: -200px;
        margin-left: -200px;
}
```

```
#leftRail{
        float:left;
        width:200px;
        margin-left:-200px;
        position:relative;
}
#center{
        float:left;
        width:100%;
        margin-right:-100%;
}
#rightRail{
        float:right;
        width:200px;
        margin-right:-200px;
        position:relative;
}
```

В body используются отступ, который является шириной боковых колонок. Оставшееся место — это ширина, до которой может увеличиваться блок container. В блоке container установленны границы и бордюры одинаковой ширины, но с разными знаками. Это накладывает границы поверх отступов в body, расставляя всё на свои места.

Обратите внимание, что этот метод работает только с колонками фиксированной ширины, поскольку только Opera понимает ширину границ в %-х.



Мы имеем многоколоночные макеты с колонками равной высоты, фиксированные и тянущиеся центральные блоки, чистая разметка CSS. И все, что нужно было для этого сделать, так это немного поразмыслить за пределами блока.

Указания по выполнению

Задание. Создание web-сайта. Тема выбирается в соответствии с номером по списку (см. Приложение), также возможно создание сайта по следующим тематикам (на выбор):

- Персональный сайт
- Сайт учебного заведения
- Сайт о своей группе
- Сайт компании (произвольный выбор)

Страницы сайта должны обязательно содержать следующие элементы:

- список;
- заголовки;
- таблицы;
- изображения;
- ссылки;
- текст, разбитый на параграфы;
- выравнивание текста;
- изменение цвета и размера шрифта;
- формы.

Ваша задача состоит в том, чтобы сформировать макет сайта не с помощью Html таблиц, а с помощью элементов div.

Спланируйте сайт в виде 1 главной и нескольких вспомогательных страниц (см. теоретическую часть).

Выполните верстку сайта с помощью блочной верстки. Студенту необходимо воспроизвести разметку, которая задана в приложении Г, воспользовавшись для этого блоками div.

Разрешается использовать только статическую и относительную верстку (значения которые может принимать свойство position={static, relative, ~~absolute, fixed~~}), допускается использовать возможность обтекания блоков (свойство float), различные способы отображения блоков (свойства display), управлять порядком наложения слоев с помощью свойства z-index.

В ходе выполнения работы необходимо:

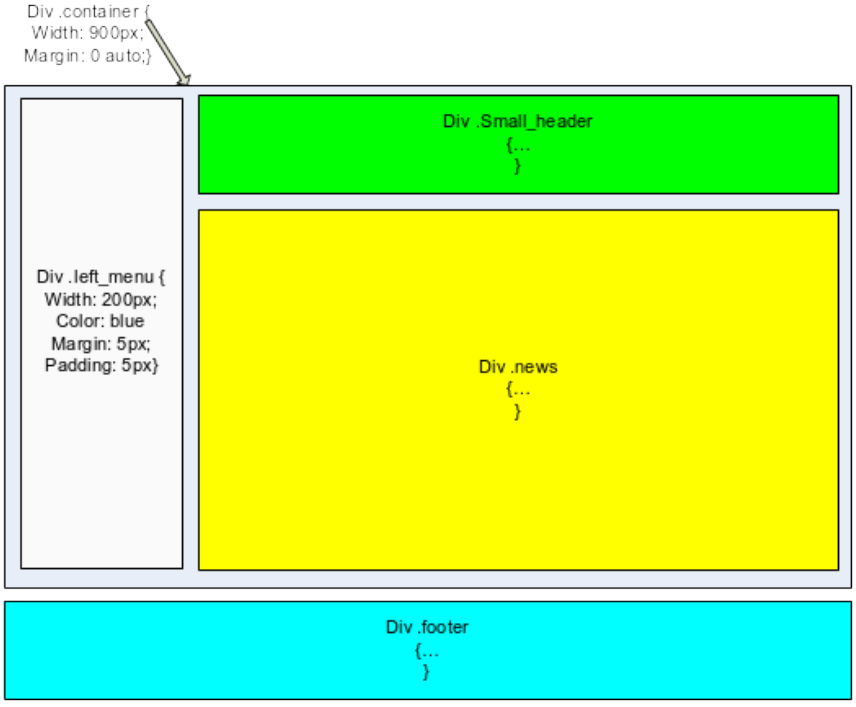
1. Разработать макет и структуру сайта.
2. Задать на страницах сайта элементы необходимые элементы (по вариантам в соответствии с приложением)
3. Создать стилевой файл style.css. Описать в этом файле css-правила для создания разметки и внешнего вида элементов:

- Используйте свойства шрифта, цвета, фона, текста для оформления элементов (тегов).
- Используйте свойства блоков для оформления нескольких блоков (весь документ, рисунок, таблица, и т.д.).
- Оформите границы блоков в соответствии с таблицей 1
- С помощью наложения слоев добавьте подпись одного из изображений, разметив ее поверх картинки.
- Оформите нумерованные и ненумерованные списки с помощью стилей.
- Воспользуйтесь псевдоклассами для оформления отдельных элементов (ссылки, первая строка абзаца, первая буква абзаца и т.д.).
- Создайте стилевой файл с альтернативным оформлением (сохраните его с тем же именем, но в другой папке, чтобы можно было применить либо первый, либо второй стилевой файлы).

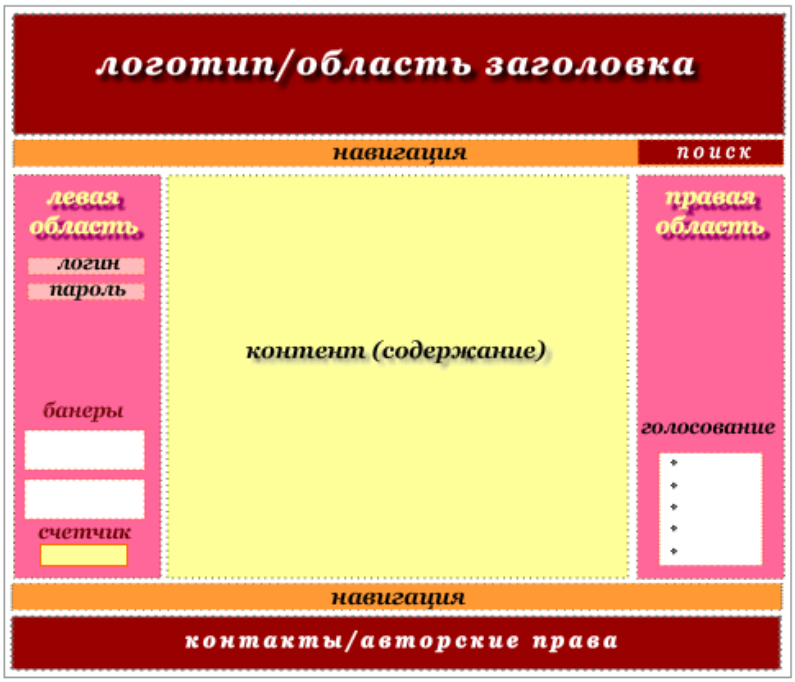
Таблица 1 Стиль оформления границы блоков

№ варианта	Тип границы
1	Dashed
2	Dotted
3	Solid
4	Double
5	Groove
6	Ridge
7	Outset
8	Inset

Перед началом верстки, разработаете блочный макет сайта, примерные варианты смотрите на рисунке ниже. Макет состоит из вложенных друг в друга блоков div с описанием их свойств.



или



Блочный макет сайта.

Создайте на отдельной странице формы заданного вида (приложение 2). При верстке форм используйте таблицы. Форма должна содержать все возможные поля ввода.

В местах где подразумевается ввод нескольких позиций, предусмотреть размещение двух строк (позиций). Определить поля подстановки и информацию, которая в них заносится. Реализовать форму средствами HTML.

Номера вариантов выбираются в соответствии с номером компьютера. Варианты документов приведены в приложении 2.

Кроме того, создайте форму с использованием HTML5 по образцу:

Исходный HTML код

```
<form>
 Число из интервала:<br/>
 <input type='range' min='0' max='100' step='10' value='10'
 Целое число:<br/>
 <input type='number' min='-100' max='100' step='1' value='
 Дата и время:<br/>
 <input type='datetime'/><br/>
 Дата:<br/>
 <input type='date'/><br/>
 Время:<br/>
 <input type='time'/><br/>
 Месяц:<br/>
 <input type='month'/><br/>
 Неделя:<br/>
 <input type='week'/><br/>
 Цвет:<br/>
 <input type='color'/><br/>
 url:<br/>
 <input type='url' value='domain.ru'/><br/>
 email:<br/>
 <input type='email' value='@domain.ru'/><br/><br/>
 <input type='submit' value='Подтвердить'/>
</form>
```

Число из интервала:

Целое число: 12

Дата и время: щхжз

Дата: 29.мм.гггг

Время: --:--

Месяц: Декабрь ---- г.

Неделя: Неделя --, ----

Цвет:

url: domain.ru

email: @domain.ru

Подтвердить

```
<div><meter min='-50' low='-10' high='30' max='50' value='-15' title='градусы'></meter></div>
<div><meter min='-50' low='-10' high='30' max='50' value='20' optimum='20' title='градусы'></meter></
<div><meter min='-50' low='-10' high='30' max='50' value='35' title='градусы'></meter></div>

<div><progress max='100' value='70' title='%'></progress></div>
```

Размещение сайта на веб-сервере

После разработки и отладки веб-сайта на локальном сервере, он должен быть размещен в сети Интернет. Обычно для этого используетсмя виртуальный хостинг, поддерживающий доступ к серверу через административный веб-интерфейс и/или по протоколу ftp.

Если вы решили поместить сайт на сервере своего провайдера, то вам следует получить у него информацию, необходимую для доступа:

• выделенный вам адрес URL. Возможно, провайдер даст вам возможность самому выбрать адрес;

• логин или имя для доступа;

• пароль для доступа.

Кроме того, вам необходимо выяснить, по какому протоколу осуществляется доступ к серверу: HTTP или FTP.

Следующее, что нужно сделать - это зарегистрироваться в бесплатной службе. Процедура регистрации очень простая. Рассмотрим кратко для примера, как зарегистрироваться на сервере **www.narod.ru**.

Установите соединение с Интернетом и загрузите начальную страницу сайта **www.narod.ru**, указав этот адрес в адресной строке браузера.

В левом верхнем углу страницы есть ссылка **Регистрация**. Щелкните на ней мышью. На экране появится форма , которую следует заполнить.

В форме вы должны указать выбранное вами имя для вашего сайта, логин и пароль для доступа, а также персональные данные: фамилию, имя, адрес, E-mail и др.

При заполнении формы следует строго придерживаться правил, установленных бесплатной службой. Эти правила указаны здесь же на форме. Например, на данном сервере требуют, чтобы пароль содержал не менее четырех символов и не совпадал с логином. Поля, обязательные для заполнения, помечены звездочкой *.

Заполнив форму, нажмите кнопку **Зарегистрироваться** в конце формы. Если вы все ввели правильно, то появится страница с сообщением об успешной регистрации.

Возможно также, что имя, которое вы выберите для сайта, окажется занятым. Если это произойдет, то вы получите соответствующее сообщение с предложением выбрать другое имя.

Когда регистрация будет закончена, прервите связь с Интернетом.

Через некоторое время на адрес E-mail, который был указан при регистрации, вы получите письмо с подтверждением регистрации и основными регистрационными данными: адресом сайта, адресом E-mail, логином и паролем. Далее в письме будет приведена информация об основных сервисах и, в частности, указано, что для загрузки файлов на сервер следует использовать либо интерфейс загрузки, т.е. файловый менеджер, либо FTP-доступ.

Теперь, когда вы зарегистрировали свой сайт и получили всю необходимую для доступа информацию, можно приступать к пересылке файлов. Посмотрим, как для этой цели использовать интерфейс загрузки сервера **www.narod.ru**.

Установите связь с Интернетом и загрузите главную страницу **www.narod.ru**.

В правой части этой страницы в группе элементов управления **Вход** введите в соответствующих полях ввода ваши **Логин** и **Пароль**.

Будьте внимательны при вводе пароля, так как вводимые символы отображаются звездочками *. Помните также, что пароль чувствителен к регистру, т.е. заглавные и строчные буквы в пароле различаются.

Нажмите кнопку **Войти**. На экране появится страница **Мастерская** .

Выберите в меню этой страницы ссылку **Загрузка файлов на сайт**. В этом разделе менеджер файлов предоставляет вам возможность выбрать файлы для загрузки.

Нажмите левую верхнюю кнопку **Обзор…** . На экране появится диалог **Выбор файла**, который ничем не отличается от стандартного диалога открытия файла.

Перейдите в папку **Web**, в которой вы сохранили файлы сайта, и щелчком мыши выберите файл index.html.

Нажатием кнопки **Открыть** закройте диалог **Выбор файла**. Полное имя выбранного файла появится в верхнем левом поле ввода, слева от кнопки **Обзор…** .

Выберите остальные файлы сайта – **licey1548.html**, **spisok.html**, **menu.html**, **МИР.JPG** - нажимая поочередно каждую следующую кнопку **Обзор…** .

Когда все пять файлов будут выбраны, нажмите кнопку **Загрузить файлы**. Через некоторое время, после того как файлы будут загружены, появится страница с сообщением об этом, а затем вы возвратитесь к странице загрузки, в верхней части которой увидите список загруженных файлов.

Таким образом, вы одновременно можете загрузить до 10-ти файлов. Если требуется переслать на сервер большее количество файлов, то операцию загрузки следует повторить.

С помощью кнопок на странице **Управление файлами** вы можете **Создать новый файл** или папку, **Редактировать**, **Скопировать**, **Переместить**, **Переименовать**, **Удалить файл** и **Изменить позицию в каталоге**. Подсказку о назначении каждой кнопки вы увидите, если установите на ней указатель мыши. Для выполнения перечисленных операций с одним или несколькими файлами, необходимо предварительно выбрать их, установив флажок слева от имени.

Несмотря на то, что стандартной кодировкой русскоязычного текста для Web-страниц является кодировка **Win1251**, некоторые бесплатные службы используют на своих серверах другие кодировки. По данным статистики русскоязычная часть Интернета использует на 95% кодировку **Win1251** и на 5% - кодировку **КОИ-8**. Так, например, на сервере **www.chat.ru** используется кодировка **КОИ-8**. Это, однако, не создает для пользователей никаких препятствий или проблем, если загрузка выполняется с помощью файлового менеджера. При пересылке ваших Web-страниц на этот сервер менеджер файлов службы спросит у вас, в какой кодировке составлены документы. Если вы использовали текстовый редактор в Windows, то кодировка документа **Win1251**, если в DOS, то кодировка **DOS**, если в Unix, то кодировка **КОИ-8**, если в Macintosh, то кодировка **Mac**. На основании вашего ответа файловый менеджер автоматически перекодирует ваши страницы в **КОИ-8**.

В том случае, когда вы создаете свои страницы в кодировке, отличной от **Win1251** и размещаете их на сервере, где явно не указана требуемая кодировка, вы должны сами перекодировать их в **Win1251** с помощью одной из специально для этого предназначенных программ.

Разные браузеры по-разному отображают Web-документы. Поэтому рекомендуется просматривать их в разных браузерах. В результате вы должны решить, какие изменения следует сделать в документах, чтобы они наилучшим образом отображались в разных браузерах, при различном разрешении экрана и разных размерах шрифта.

Иногда при загрузке Web-страниц могут возникнуть некоторые проблемы. Если ваша страница, находящаяся на сервере **www.narod.ru**, не загружается, и в окне браузера появляется примерно такое или похожее сообщение: «**Извините, запрошенный Вами документ не найден. Попробуйте начать с главной страницы**» или «**Невозможно отобразить страницу**» (The page cannot be displayed), то это значит, что адрес сайта введен вами неправильно. Проверьте указанный вами адрес. Если же появляется сообщение «**Не найден файл index.html или доступ к каталогу запрещен**», то это означает, что страница не помещена на сервер или среди посланных вами файлов

отсутствует файл **index.html**. Содержащаяся в этом файле страница всегда выводится как первая.

Регистрация в поисковых системах и каталогах

Ваш сайт создан и размещен на сайте провайдера. Хотя теперь он и стал доступен всему миру, это не значит, что весь мир сразу узнает о нем и отправится с ним знакомиться. Для того чтобы люди узнали о вновь появившемся сайте и стали посещать его регулярно, необходимо, прежде всего, зарегистрировать его в поисковых машинах и каталогах.

Когда вы ищете информацию в Интернете, вы, скорее всего, обращаетесь к поисковым системам, таким как **Yandex**, **Rambler**, **Altavista**, или к Интернет-каталогам, например, **Aport**, **Mail.ru**, **Yahoo**. Поэтому очень важно, чтобы данные о вашем сайте попали в подобные системы, и каждый желающий, который интересуется тематикой вашего сайта, мог его легко обнаружить.

Процесс попадания информации о вашем сайте в каталоги и поисковые системы, иногда называемые поисковыми машинами, состоит из нескольких этапов.

• Вы вводите данные и описание своего сайта на специальной странице Интернет-каталога или поисковой машины.

• Специальная программа-робот поисковой машины обращается по указанному вами адресу и просматривает все страницы вашего сайта, анализируя все встреченные на страницах слова, особо отмечая для себя название каждого документа, его описание, заголовки, встречающиеся в тексте, а затем следует дальше по ссылкам к другим документам. Этот процесс называется **индексацией**. Программа-робот вернется на ваш сайт через некоторое время, например, через месяц, чтобы проиндексировать его заново. Если же вы зарегистрировались в Интернет-каталоге, то, как правило, функцию проверки осуществляет человек - сотрудник каталога. Он проверяет, существует ли сайт по указанному вами адресу и соответствует ли он данному вами описанию. Иногда проверяющий сам составляет описание для вашего Web-сайта.

• Данные о сайте, который вы зарегистрировали, появляются в поисковой машине или Интернет-каталоге. Введя в строку поиска слова, которые встречаются на вашем сайте, в результатах поиска вы можете увидеть ссылку на соответствующие страницы вашего ресурса. В зависимости от популярности поисковика, на котором вы регистрируетесь, от внесения данных до появления информации может пройти от 1 дня до двух недель.

Как вы могли заметить по собственному опыту, к некоторым поисковым системам вы обращаетесь каждый день, а к некоторым - время от времени. Большинство пользователей Интернета поступает точно так же. Поэтому есть сайты, на которых регистрироваться нужно обязательно, чтобы не потерять своих потенциальных посетителей. В остальных поисковых системах и каталогах можно регистрироваться, если у вас есть для этого свободное время или воспользоваться системами, регистрирующими ваш ресурс сразу в нескольких поисковиках. На популярных поисковых системах лучше регистрироваться вручную.

Вот список популярных поисковых систем, в которых нужно обязательно зарегистрировать свой сайт:

• **Яндекс** (***http://www.yandex.ru***)
• **Rambler** (***http://www.rambler.ru***)
• **Апорт** (***http://www.aport.ru***)
• **Altavista** (***http://www.altavista.com***)

Из каталогов наиболее популярными являются следующие:

• **Mail.RU** (**http://www.mail.ru**)
• **Интернет-столица** (**http://www.data.ru**)
• **Yahoo** (**http://www.yahoo.com**)

Сейчас все большее число поисковых систем включают в себя также и каталог, так что регистрируя сайт в поисковой системе, вы можете одновременно включить его и в каталог при ней.

Регистрация в каталогах наиболее проста: все, что от вас потребуется - это правильно подобрать раздел, в котором должно находиться описание вашего сайта. В некоторых каталогах не нужно и этого - сотрудник каталога сам разместит описание вашего сайта в нужном разделе, а то и в нескольких.

Подготовка к регистрации в автоматических поисковых системах потребует небольшой корректировки файлов вашего Web-сайта.

Поисковые системы в ответ на запрос, как правило, выдают сотни страниц, в которых содержатся введенные пользователем слова, но отображают на экране одновременно не более 10-20 заголовков, помещая вперед те, которые, по мнению системы, наиболее подходят для ответа на запрос. Большинство пользователей просматривает не более двух-трех таких партий заголовков, оставляя остальные без внимания. Поэтому необходимо попытаться сделать так, чтобы страницы вашего сайта, содержащие запрошенные слова, были помещены поисковой системой как можно выше в выдаваемом ею списке.

Перед началом регистрации в поисковых системах имеет смысл еще раз просмотреть все страницы вашего сайта, чтобы проверить - правильно ли составлены заголовки и хорошо ли в них отображен смысл следующего за ними текста, как часто встречаются в тексте страницы слова, которые можно счесть ключевыми при описании данной страницы.

Также внимательно просмотрите заголовки страниц, которые отображаются в строке заголовка браузера (на самом верху окна браузера). Они должны соответствовал содержанию страницы. Такой заголовок в коде странице находится между тэгами **<title>** и **</title>** и его легко найти и отредактировать. Это можно сделать либо с помощью вашей программы редактирования Web-страниц, либо вручную с помощью текстовок редактора.

Сам процесс регистрации в поисковой системе довольно прост и не отнимет у вас много времени. Рассмотрим регистрацию в одной из самых популярных российских поисковых систем **Яндекс**. Регистрация в других системах производится аналогично.

Наберите в адресной строке вашего браузера адрес поисковой системы, например, **http://www.yandex.ru**. Загрузится главная страница поисковой системы.

Найдите на странице ссылку **Добавить сайт**, обычно расположенную в нижней части главной страницы системы, и нажмите на нее. В разных системах эта ссылка может называться по разному, например. **Добавить ресурс**, **Добавить URL**, **Add URL**, **Submit a site** и так далее. После нажатия на ссылку откроется страница добавления сайта в поисковую систему.

Прочитайте правила регистрации в системе и заполните регистрационную форму.

Нажмите кнопку **Добавить URL!**, и через несколько секунд загрузится страница с сообщением об успешной регистрации.

Система **Яндекс** также сразу предлагает внести информацию о вашей странице в каталог. Для внесения в каталог информации о вашем сайте заполните поля новой регистрационной формы.

Нажмите кнопку **Добавить!**, и через несколько секунд появится сообщение о том, что ваш ресурс будет добавлен в каталог после проверки гидом.

Внесите название, адрес поисковой системы и дату регистрации в таблицу, которую вы завели для учета подобной информации.

На этом регистрацию в поисковой системе можно закончить и перейти к регистрации в других системах, а можно предпринять еще несколько дополнительных шагов.

Обычно достаточно добавить в поисковую систему адрес главной страницы вашего сайта и поисковый робот сам пройдет по остальным страницам сайта и проиндексирует

их. Тем не менее, быстро проиндексировав главную страницу, для индексации остальных страниц робот может вернуться на ваш сайт только через несколько недель. Поэтому имеет смысл также отдельно зарегистрировать наиболее важные страницы вашего сайта.

Выше были перечислены поисковые системы, в которых обязательно нужно зарегистрироваться вручную. Регистрацию вашего Web-сайта на остальных поисковых системах можно поручить специальным регистрационным службам.

Услуги бесплатной регистрации предоставляют следующие службы:

• **TAU** (**http://www.design.ru/free/addurl/**) - регистрирует сайт во всех основных российских и зарубежных поисковых системах.

• **AddMe!** (**http://www.addme.com/**) - регистрирует в 25 зарубежных поисковых машинах

• **@Submit!** (**http://www.uswebsites.com/submit/**) - регистрирует в основных зарубежных поисковых системах.

Через некоторое время имеет смысл проверить, правильно ли робот поисковой машины проиндексировал ваш сайт. Для этого скопируйте какую-либо фразу, встречающуюся на одной из страниц вашего Web-сайта, и поместите ее в качестве запроса в поисковую систему. Страница вашего сайта должна быть вверху списка найденных страниц. Если этого не происходит, то ваш сайт не был проиндексирован или был проиндексирован неправильно.

Большинство поисковых систем позволяет искать ключевые слова только на определенном сервере. Для дополнительной проверки индексации можно задать поиск ключевых слов только в рамках своего сайта. Обычно подобный поиск можно осуществлять на специальной странице поисковой системы, на которую можно перейти с главной страницы системы, нажав ссылку **Расширенный поиск**, **Поиск с языком запросов**, **Advanced search** или подобную.

Если страницы вашего сайта оказываются не проиндексированными в поисковой системе более двух недель, имеет смысл повторите регистрацию в данной системе.

Задания к работе

1. Определить необходимые и достаточные требования к хостингу, позволяющие обеспечить функционирование веб-сайта.
2. Используя поисковую систему найти веб-сайты, представляющие сводную информацию, обзоры и сравнение хостинговых площадок.
3. По результатам анализа выбрать хостинг, соответствующий сформированным требованиям и зарегистрироваться там.
4. Выбрать и зарегистрировать доменное имя для вашего сайта.
5. Разместить сайт на сервере хостинг-провайдера.

Содержание отчета

1. Цель
2. Ход работы
3. Обоснование выбора структуры сайта
4. Схема логической структуры созданного сайта.
5. Схема физической структуры созданного сайта (до уровня файлов)
6. Словесная постановка задачи лабораторной работы.
7. Последовательность выполнения работ в среде NotePad++ или Блокнот для создания сайта.
8. Макетирование сайта с помощью элементов div.
9. Формализация выполненного индивидуального задания (разделение структуры и оформления создаваемых документов).
10. Выводы

Контрольные вопросы

1. Определите основные принципы разработки сайта.
2. Перечислите наиболее распространенные виды логических структур веб-сайтов
3. Сравнительная характеристика логической и физической структуры сайта.
4. Почему в именах файлов нежелательно использование символов русского алфавита
5. Сформулируйте принцип организации, преимущества, недостатки и область применения решетчатой структуры сайта.
6. Макетирование сайта. Блочный элемент Div. Фиксированный и резиновый дизайн.
7. Размещение сайта в сети Интернет.
8. Регистрация сайта в поисковых системах и каталогах.
9. Формы. Элементы формы. (см. Учебное пособие.)

Приложение 1

Таблица1.1. Тематика сайтов

№	Тема	№	Тема	№	Тема
1	Цветы	9	Бытовая техника	17	Животные
2	Детские игрушки	10	Книги	18	История
3	Актеры	11	Компьютерные игры	19	Музыкальные группы
4	Писатели	12	Автомобили	20	Спорт
5	Программы	13	Города	21	Путешествия
6	Операционные системы	14	Страны	22	Картины
7	Мебель	15	Фрукты	23	Самолеты
8	Недвижимость	16	Монеты	24	Космос

Таблица 1.2. Обязательные элементы

№	Схема
1	Заголовок, список нумерованный, 2 параграфа, заголовок, горизонтальная линия1, параграф, заголовок, неформатированный текст, горизонтальная линия2, список ненумерованный
2	Заголовок, горизонтальная линия2, список ненумерованный, 2 параграфа, заголовок, параграф, неформатированный текст, горизонтальная линия1, заголовок, список нумерованный
3	Заголовок, горизонтальная линия1, список нумерованный, параграф, заголовок, 2 параграфа, заголовок, неформатированный текст, список ненумерованный, горизонтальная линия2
4	Список ненумерованный, заголовок, неформатированный текст, список нумерованный, заголовок, параграф, заголовок, горизонтальная линия1, 2 параграфа, горизонтальная линия2
5	Список нумерованный, заголовок, неформатированный текст, 2 параграфа, горизонтальная линия1, заголовок, параграф, заголовок, горизонтальная линия2, список ненумерованный
6	Горизонтальная линия2, заголовок, список ненумерованный, 2 параграфа, заголовок, 2 параграфа, заголовок, неформатированный текст, горизонтальная линия1, список нумерованный
7	Горизонтальная линия, заголовок, 2 параграфа, заголовок, параграф, заголовок, неформатированный текст, список ненумерованный, горизонтальная линия, список нумерованный
8	Заголовок, горизонтальная линия1, список нумерованный, 2 параграфа, заголовок, 2 параграфа, неформатированный текст, горизонтальная линия2, заголовок, список ненумерованный
9	Неформатированный текст, заголовок, горизонтальная линия2, список ненумерованный, параграф, заголовок, параграф, горизонтальная линия1, заголовок, список нумерованный, параграф
10	Неформатированный текст, заголовок, 2 параграфа, заголовок, горизонтальная линия1, список ненумерованный, параграф, горизонтальная линия2, заголовок, список нумерованный

Таблица 1.3 Стили оформления текста

Параметр	Вар 1	Вар 2	Вар 3	Вар 4	Вар 5	Вар 6	Вар 7	Вар 8	Вар 9	Вар 10
Нумерованный Список:										
Маркер	1	i	I	a	A	i	a	A	I	1
Номер перв ел	5	3	1	2	10	20	2	5	10	-5
Уровень вложен-ти	2	3	4	2	3	4	2	3	4	2
Цвет	Синий	Крас-ный	Голу-бой	желтый	Чер-ный	Бе-лый	Зеле-ный	Розо-вый	Оран-жевый	Сала т-ный
Ненумерованный список										
Маркер	disk	circle	square	disk	circle	square	disk	circle	square	disk
Уровень вложен-ти	4	2	3	4	2	3	4	2	3	4
Цвет	Салат ный	Си-ний	Крас-ный	Голу-бой	Жел-тый	Чер-ный	Бе-лый	Зеле-ный	Розо-вый	Оран-жевый
Заголовки										
Цвет	Крас-ный	Голу-бой	Жел-тый	Чер-ный	Бе-лый	Зеле-ный	Розо-вый	Оран-жевый	Салат ный	Синий
Размер	2	3	4	5	6	5	4	3	2	6
Выравн ив	left	center	right	left	center	right	left	center	right	center
Параграф										
Выравн ив	center	right	left	center	right	left	center	right	center	left
Размер текста	Big	smal	Medium	big	small	big	Medium	Small	Big	Medium
Шрифт	Arial	Tahoma	Courier	Times New R	Arial	Tahoma	Courier	Times New R	Arial	Tahoma
	жирн	курсив	подчерк	жирн	курсив	подчерк	жирн	курсив	подчерк	жирн
Страница										
Фон	Бе-лый	Жел-тый	Чер-ный	Рису-нок	Крас-ный	Голубой	Рису-нок	Чер-ный	Темно сини й	белы й
Цвет текста	Си-ний	Красный	красный	Оранжевый	Зеленый	синий	зеленый	Жел-тый	белый	салатный
Горизонтальная линия1										
Высота	2	5	10	20	10	30	2	20	30	5

Выравнив	center	right	left	center	right	left	center	right	left	center
ширина	100%	50%	200px	100px	30%	50%	100 %	400px	300px	70%
Цвет	Крас-ный	Голу-бой	желтый	Чер-ный	Бе-лый	Зеле-ный	Розо-вый	Оранжевый	Салат-ный	Синий
Обьем	+	-	+	-	+	-	+	-	+	-
Горизонтальная линия2										
Высота	5	2	20	10	30	10	20	2	5	30
Выравнивание	right	center	center	left	left	right	right	center	center	left
ширина	50%	100 %	100px	200px	50%	30%	400px	100%	70%	300px
Обьем	-	+	-	+	-	+	-	+	-	30

Возможные значения параметры ячеек таблицы описаны в таблице 1.4.

Таблица 1.4. Описание параметров ячеек таблицы

Параметр	**Значение**	**Описание**
Ширина	Ф - 200	Ширина ячейки фиксированная, 200 пикселей
	ФП-50	Ширина ячейки фиксированная, 50% ширины контейнера
	Н	Ширина не задана, определяется содержимым
	М	Максимальная ширина ячейки
Содержимое	МВ-5	Вертикальное меню, в виде списка из 5 элементов
	МГ-3	Горизонтальное меню в виде набора ссылок из 3 элементов
	З	Заголовок
	П	Параграф
	ЗП	Заголовок и параграф
	А	Автор и дата последнего обновления
	И	Изображение
	И-4	4 упорядоченных изображения
	Е	e-mail
Фон	FF00FF	Цвет фона
	img	Фон задается изображением
	-	Нет фона
	A0F	Цвет в 16ом формате, сокращенная запись для AA00FF

Варианты заданий определяются номером студента по списку. Контент, предлагается студентом. Варианты структуры таблицы и свойства ее ячеек задаются таблицей 1.6

Параметры таблицы задаются таблицей 1.5.

Таблица 1.5 Расположение и размер таблицы

Вариант	**Ширина**	**Выравнивание**	**cellpadding**	**Cellspacing**	**Граница**
1	ФП-100	-	5	3	2
2	Ф-600	Центр	-	10	1
3	Ф-800	Left	-	5	5
4	М	Right	2	4	2
5	ФП-100	-	3	-	4
6	ФП-80	Центр	-	-	3
7	М	Left	5	2	2
8	Ф-800	Right	4	3	2
9	Н	-	3	4	3
10	ФП-50	Центр	-	5	4

Таблица 1.6 Варианты заданий

№	Задание	№ ячеек							
1	Таблица	1	2	3	4	5	7	9	13
	Ширина	Ф200, ФП50, Н							
	Содержимое	2хИ, А, Е, МГ-5, ЗП, П, МВ-6							
	Фон	#ABC	#00D	#D00	#ABC	img	#ABC	img	#ABC
2	Таблица	2	5	8	9	10	11	12	13
	Ширина	Ф300, ФП30, М							
	Содержимое	И, А, Е, МГ-3, ЗП, 2хП, МВ-4							
	Фон	img	#CDE	#DEF	#AAA	img	#BCE	img	img
3	Таблица	1	4	7	9	10	13	14	15
	Ширина	Ф200, ФП50, М							
	Содержимое	И, А, Е, 2хМГ-2, ЗП, П, МВ-5							
	Фон	#DDE	img	img	#CDE	#BBC	img	#DED	#ACB
4	Таблица	1	3	4	7	8	13		
	Ширина	Ф200, М, Н							
	Содержимое	И, А, Е, МГ-2, ЗП, П, 2хМВ-3							
	Фон	#FCE	#DCE	#CCC	img	img	#ADC	img	img
5	Таблица	1	4	6	9	12	13	14	15
	Ширина	М, ФП50, Н							
	Содержимое	И-4, А, Е, МГ-2, ЗП, П, МВ-5							
	Фон	#DDE	img	img	#DDE	img	img	#CDE	#BBC
6	Таблица	1	3	5	7	9	10	12	13
	Ширина	Ф200, ФП25, Н							
	Содержимое	И, А, Е, МГ-4, 3х3П, П, МВ-2							
	Фон	img	#FCE	#AAA	img	#BBB	#CDE	#AED	img
7	Таблица	3	4	7	8	9	10	12	15
	Ширина	Ф200, ФП50, Н							
	Содержимое	2хИ, А, Е, МГ-3, ЗП, П, МВ-5							
	Фон	#DDC	#BCE	img	img	img	#EDA	img	#FAB
8	Таблица	2	3	5	6	9	12	14	15
	Ширина	Ф240, ФП40, М							
	Содержимое	И, А, Е, МГ-4, ЗП, 3хП, МВ-5							
	Фон	#DCC	img	#CDE	#AAC	#ABC	#FEC	img	img
9	Таблица	1	3	4	5	6	10	12	13
	Ширина	Ф250, ФП50, Н							
	Содержимое	И-5, А, Е, МГ-2, ЗП, П, МВ-4							
	Фон	img	img	#DEE	#AAC	#ADA	img	#EDB	#BBD
10	Таблица	1	2	3	5	7	10	11	13
	Ширина	ФП40, М, Н							
	Содержимое	И, А, Е, 2хМГ-2, ЗП, П, МВ-5							º
	Фон	#ABC	#AAF	img	img	#DDE	#BCE	#CCE	img

11	Таблица	1	2	4	5	7	9	12	15
	Ширина	Ф200, ФП50, Н							
	Содержимое	3хИ, А, Е, МГ-2, ЗП, П, МВ-2							
	Фон	img	#BCE	img	#DDF	#ACA	img	#ABA	img
12	Таблица	1	3	7	9	10	11	13	15
	Ширина	М ФП50, Н							
	Содержимое	И, А, Е, МГ-4, ЗП, П, 3хМВ-5							
	Фон	#ADA	img	#CCC	img	#DDD	#BCB	img	#ADE
13	Таблица	3	4	7	9	10	11	12	14
	Ширина	Ф200, ФП50, Н							
	Содержимое	И, А, Е, МГ-3, 2хЗП, П, МВ-5							
	Фон	img	#CDE	#FFA	img	img	#AAF	#BCE	img
14	Таблица	1	2	3	4	5	6	10	12
	Ширина	Ф250, ФП40, Н							
	Содержимое	И, А, Е, 2хМГ-2, ЗП, П, МВ-5							
	Фон	#CEA	#FED	img	#ACA	img	#DCE	#BBC	img
15	Таблица	2	3	5	6	7	8	13	14
	Ширина	Ф300, ФП50, М							
	Содержимое	И, А, Е, 2хМГ-2, ЗП, П, 2хМВ-3							
	Фон	#BCA	img	#CDA	#ACE	img	#DDD	img	img

Пояснения к таблице:

1. Если в качестве фона указано img, то это означает использовать любую фоновую картинку.
2. Для каждого варианта есть обязательный набор содержимого (расшифровку см. в таблице 1.4). Если перед обозначением указан множитель (например, 2хИ), то это говорит о том, что данный элемент должен быть размещен указанное число раз. Место для размещения элементов выбирается студентом самостоятельно, исходя из получившейся после упрощения таблицы.
3. Студент самостоятельно выбирает к каким ячейкам таблицы следует применить параметры ширины (расшифровку см. в таблице 1.4)

Приложение 2

Таблица 2.1. Элементы формы

Обознач.	Описание
ТП	Текстовое поле, элемент INPUT. ТП-20 — текстовое поле с SIZE=20.
ТБ	Текстовый блок, элемент TEXTAREA. ТБ-64-6 — текстобый блок с COLS=64 ROWS=6.
ВС	Выпадающий список, элемент SELECT. ВС-3 — выпадающий список из 3х элементов.
МВС	Список с множественным выбором, элемент SELECT. МВС-3 — список из 3х элементов.
РК	Радиокнопка, элемент RADIO. РК-3 — три радиокнопки.
ФЛ	Флажок, элемент CHECKBOX. ФЛ-3 — три флажка.
ПР	Пароль, элемент PASSWORD
СБ	Кнопка сброса, элемент RESET
ОК	Кнопка отправки, элемент SUBMIT. СБ-2 — две кнопки отправки.
ПК	Картинка CAPTCHA. Комбинация из любой картинки размером 400x40 пикселей и поля INPUT.
ПС	Проверочный ответ на вопрос. Комбинация из текста вида «1+2=?» и поля INPUT.
ЗГ	Загрузка файла, элемент INPUT с аттрибутом FILE.

Таблица 2.2. Варианты заданий:

№	Описание формы
1	Форма обратной связи. Информация: от кого, почтовый адрес, сообщение, проверочный код, тема сообщения, оценка сайта.
2	Гостевая книга. Информация: тема сообщения, от кого, сайт, сообщение, проверочный код, подписка на обновление гостевой книги.
3	Форма жалобы. Информация: вид жалобы, почтовый адрес, текст жалобы, подписка на новости сайта, проверочный код.
4	Форма заказа товара. Информация: товар, свойство товара (например: цвет), ФИО заказчика, адрес доставки, подписка на новости сайта.
5	Форма отзыва. Информация: вид отзыва, от кого, сайт, сообщение, проверочный код, подписка на новости сайта.
6	Форма голосования. Информация: выбор варианта голосования, почтовый адрес, комментарий к голосу, проверочный код.
7	Форма подачи доклада на конференцию. Информация: ФИО, выбор секции, файл доклада, проверочный код.
8	Форма поиска по сайту. Информация: выбор раздела сайта для поиска, поисковый запрос, точное совпадение, искать по раздельным словам.
9	Форма регистрации на сайте. Информация: логин, пароль, почтовый адрес, комментарий к регистрации, проверочный код.
10	Форма формирования отчета. Информация: выбор типа отчета, поле ввода даты, выбор полей для отчета, выбор типа сортировки.

Таблица 2.3. Варианты элементов формы:

№	Требуемые элементы
1	ПК, СБ, ОК, ТП-20, ТП-10, ТБ-40-5, ТП-20, ВС-3
2	ПС, ОК-2, ТП-10, ТП-10, ТБ-50-4, РК-2
3	ПК, СБ, ОК, ВС-3, ТП-10, ТБ-50-5, ФЛ-2
4	ПС, ОК-3, ТП-20, РК-4, ТП-20, ТП-10, ФЛ
5	ПК, СБ, ОК-2, РК-3, ТП-10, ТП-20, ТБ-40-5, ФЛ
6	ПС, СБ, ОК, МВС-4, ТП-10, ТБ-20-5
7	ПК, ЗГ, ОК-3, ТП-20, ВС-5
8	ПС, СБ, ОК, ВС-6, ТП-20, ФЛ-4
9	ПК, ПР, СБ, ОК-2, ТП-10, ТП-10, ТП-20, ТБ-20-5
10	ПС, ОК-3, ВС-4, ТП-10, ФЛ-4, РК-2
11	ПК, СБ, ОК, ТП-10, ТП-15, ТП-10, ТБ-64-5, ВС-2
12	ПС, СБ, ОК-2, ТП-20, ТП-10, ТБ-60-3, РК-4
13	ПК, ОК-2, МВС-4, ТП-20, ТБ-40-6, ФЛ-4
14	ПС, ОК-3, ТП-10, РК-6, ТП-10, ТП-20, ФЛ-2
15	ПК, СБ, ОК, РК-4, ТП-20, ТП-30, ТБ-50-6, ФЛ-2
16	ПС, СБ, ОК-2, ВС-5, ТП-20, ТБ-40-6
17	ПК, ЗГ, ОК-2, ТП-15, РК-6
18	ПС, СБ, ОК, МВС-4, ТП-15, РК-2, РК-2
19	ПК, ПР, СБ, ОК, ТП-20, ТП-20, ТП-10, ТБ-64-5
20	ПС, СБ, ОК, РК-4, ТП-15, ФЛ-5, ВС-3

Лабораторная работа №№ 15-16-17

Использование языка сценариев JavaScript при создании web-сайта

Цель: *сформировать основные умения и навыки по созданию сценариев средствами JavaScript, навыки открытия новых окон, динамического создания документов, их закрытия. Отработать навыки организации диалога с пользователем с помощью методов* ***alert(), confirm()*** *и* ***prompt()****; навыки составления программного кода с использованием условного оператора* ***if.***

Ход работы:

17. Изучить теоретический материал.
18. Выполнить примеры.
19. Выполнить задания в соответствии с указаниями.
20. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
21. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Язык программирования JavaScript был разработан Бренданом Эйком (Brendan Eich) в Netscape Communications как язык сценариев для обозревателей Netscape Navigator, начиная с версии 2.0. В дальнейшем к развитию этого языка подключилась корпорация Microsoft, чьи обозреватели Internet Explorer поддерживают JavaScript, начиная с версии 3.0. Версия Microsoft получила название JScipt, поскольку JavaScript является зарегистрированной маркой фирмы Netscape.

JavaScript - это язык программирования, язык сценариев (скриптов), предназначенный прежде всего для создания интерактивных HTML-страниц. Программу на языке JavaScript либо встраивают прямо в HTML-файл (как в секции <**head**>, так и в секции <**body**>) с помощью тега**<script> ... </script>** и располагают код JavaScript внутри этих тегов,

<script type="text/javascript"> </script>

или помещают весь код JavaScript в отдельный файл с расширением .js и связываются с ним с помощью тега Script:

<script type="text/javascript" src="scripts/JavaScriptFile.js"> </script>

Чтобы программа была выполнена, HTML-файл должен быть открыт в браузере пользователя. Так как JavaScript является в настоящее время единственным языком сценариев, который поддерживают все основные браузеры Web (Internet Explorer, Firefox, Netscape, Safari, Opera, Camino и т.д.), то он используется очень широко. Однако следует помнить, что некоторые скрипты действуют по-разному в разных браузерах (то, что работает в Internet Explorer, может не работать в Firefox)

JavaScript - *интерпретируемый* язык. Это означает, что для исполнения программы не требуется предварительная *компиляция* (преобразование исходного текста программы в машинный код). Текст программы *интерпретируется*, то есть анализируется и сразу же исполняется.

JavaScript - это объектно-ориентированный язык программирования (ООП), основан не на обработке команд кода, а на присвоении отдельным элементам программы конкретных событий и выполнении их, если данное событие имело место. Например, событие нажатие на кнопку приводит к изменению содержимого текстового поля:

Вы можете использовать в скрипте множество различных типов функций обработки событий. В большинстве случаев функции представляют собой лишь способ связать вместе нескольких команд. Функции могут также использоваться совместно с процедурами обработки событий.

Основными понятиями любого объектно-ориентированного языка являются объекты, классы, методы и свойства. Разберем основные понятия на конкретных примерах:

```
<script type="text/javascript">
document.write("Введите свое имя и нажмите кнопку")
</script>
```

В результате при просмотре данной страницы в браузере появится текст: "Введите свое имя и нажмите кнопку".

Основные понятия:

OBJECT *(объект)* - это то, с чем производится действие, событие. Это может быть документ, открываемый в окне браузера или само окно браузера, или какая-то часть документа, теги. Объект должен иметь уникальное имя (ID), чтобы к нему можно было обратиться.

В нашем случае объектом является документ HTML и к нему можно просто обратиться по имени: **document**.

Каждый объект обладает своими методами.

METHOD *(метод объекта)* - это действия, которые можно выполнять над объектом такого типа, или которые сам объект может выполнять.

Синтаксис кода: между именем объект и методом обязательно ставят **разделительный оператор точка**, после метода в скобках параметры метода.

```
Объект.Метод("параметры метода")
```

Параметры метода относятся к типу данных - **строки символов**. Строки символов нужно обязательно взять в кавычки либо в одинарные, либо в двойные.

Каждый объект обладает своими свойствами.

PROPERTY (свойство) - каждый объект имеет свои свойства. Один и тот же объект может обладать многими свойствами. Часто эти свойства необходимо изменить, при возникновении некоторого события.

Для изменения свойства объекта необходимо соблюдать следующий синтаксис:

```
Объект.свойство объекта= "новое значение свойства "
```

Например, для изменения фонового цвета документа HTML (имя данного свойства **bgColor**) следует написать следующее:

```
<script type="text/javascript">
document.bgColor='red'
</script>
```

И при просмотре в окне браузера фоновый цвет HTML документа будет красным.

Обратите внимание на то, что значение свойства **red** пишется в кавычках (одинарных или двойных), т.к. значение свойства относится к типу данных строки символов.

Разумеется, нас будет интересовать возможность изменения свойства при возникновении какого-либо события.

EVENT (событие) - это все, что случилось: открытие окна, загрузка в него документа, клик клавишей мышки или просто перемещение курсора по экрану, нажатие клавиши на клавиатуре - это все события, и они могут инициировать запуск больших и маленьких программ.

События, главным образом, инициируются теми или иными действиями пользователя. Если он щелкает по некоторой кнопке, происходит событие **Click**. Если указатель мыши пересекает какую-либо ссылку гипертекста - происходит событие **MouseOver**. Существует несколько различных типов событий.

Опишем синтаксис основных событий, использующихся в JavaScript:

onClick - Вызывается после щелчка левой кнопкой мыши на объекте. <INPUT TYPE="elementType" onClick="function">

onMouseOver - Событие происходит, когда указатель мыши помещается над гиперссылкой.<A HREF="URL" onMouseOver="function">linkText</A>

onFocus - Событие происходит в тот момент, когда пользователь переходит к элементу формы select, text или textarea для ввода данных.<INPUT TYPE="inputType" onFocus="function">

onBlur - Событие происходит в тот момент, когда элемент формы select, text или textarea теряет фокус. <INPUT TYPE="elementType" onBlur="function">

onSelect - Обработчик события onSelect вызывается в тот момент, когда выделен текст внутри элемента формы.<INPUT TYPE="textType" onSelect="function">

onSubmit - Событие происходит в момент щелчка мышью на кнопке Submit для посылки данных формы на сервер.<TAG onSubmit="function">

onLoad - Вызывается, когда загрузка документа в окно или в фрейм закончена.<BODY onLoad="function">, <FRAMESET onLoad="function">

onUnload - Вызывается, когда пользователь выходит из документа.<BODY onUnload="function">, <FRAMESET onUnload="function">

onChange - Событие происходит в тот момент, когда значение элемента формы select, text или textarea изменилось и элемент потерял фокус.<INPUT TYPE="elementType" onChange="function">

Здесь следует пояснить, что **события** (event) и **обработчики событий** (event handler) относятся к JavaScript, но они скорее «встроены» в HTML-код. Они входят в структуру документа HTML и не требуют тегов <script> и </script>. Среди разнообразных обработчиков событий для начала мы выберем один, самый популярный, — **onmouseover** (навести мышь).

Код выглядит следующим образом:

** Наведи мышь на этот текст .... **

Как уже говорилось для написания этого кода не требуются теги <script> </script>. Событие встраивается в HTML код, т.е является описанием, атрибутом тега (в данном случае тега ** **) при выполнении данного события - наведении мышкой на текст данного абзаца - изменяется свойство объекта - фон документа HTML.

И здесь есть еще одна важная особенность: **document.bgColor ='red'** нужно также записать в кавычках - одинарных или двойных. Вы можете использовать любой тип кавычек. Однако если Вы вынуждены как в данном случае ставить кавычки дважды, то можно использовать только вложенные кавычки. Не имеет значения, в каком порядке Вы использовали кавычки - сначала двойные, а затем одинарные или наоборот.

МОЖНО:

onmouseover="document.bgColor ='red' " или

onmouseover='document.bgColor ="red" '

Но НЕЛЬЗЯ:

onmouseover="document.bgColor ="red" "

onmouseover='document.bgColor ='red' '

onmouseover="document.bgColor ='red" '

А если мы хотим изменить не свойство всего документа, а только свойство какого-то абзаца? Как в данном случае мы можем изменить свойство данного объекта? Есть несколько способов.

Код данного примера (ВАЖНО! Код должен быть записан в одну строчку)

** Этот абзац меняет цвет при наведении на него мышкой с синего на красный! **

Разберем код.

1. **style="color:blue"** определяется стиль текста в данном абзаце

2. **onmouseover=** событие которое может произойти с этим абзацем, в кавычках надо указать что при этом делать

3. **this.style.color='red'** изменить стиль абзаца: цвет текста на красный:

- слово **this** используется для доступа к элементу (объекту), вызвавшему событие (к данному абзацу),
- через точку указывается его свойство **style**, дающее доступ к стилям,
- еще через точку указывается **конкретное свойство**, значение которого мы хотим изменить (**color** - цвет текста)
- затем идет знак присвоить =,
- и затем значение свойства "цвет текста" - красный (**'red'**).

Рассмотрим еще один пример. Изменение цвета фона текста:

** Цвет фона текста меняется на красный при наведении мышкой на него! **

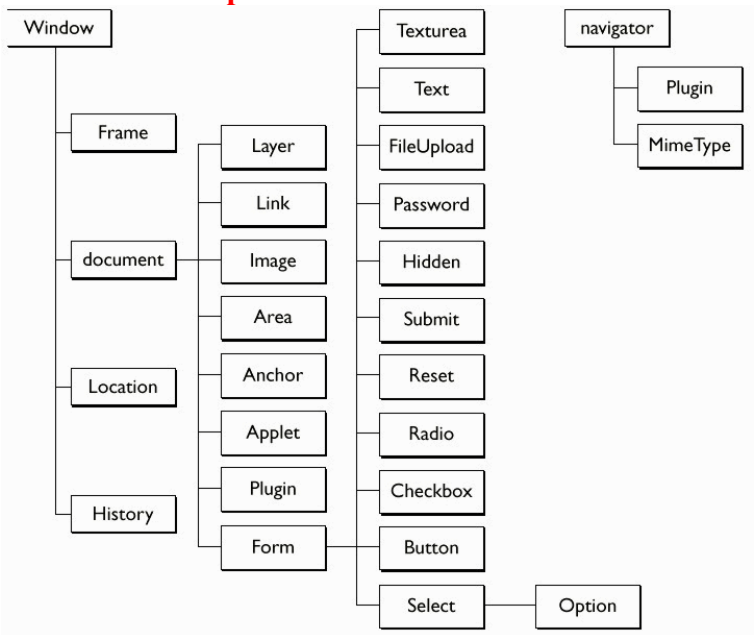
Разберем код.

1. **style="background-color:blue"** определяется стиль текста в данном абзаце (цвет фона)

2. **onmouseover=** (навести мышь) и **onmouseout=** (увести мышь) события которые могут произойти с этим абзацем, в кавычках надо указать что при этом делать

this.style.backgroundColor='red' изменить стиль абзаца: цвет фона на красный.

Иерархия объектов в JavaScript



Объект **Window** обычно соответствует главному окну браузера и является объектом верхнего уровня в языке JavaScript, поскольку документы, собственно, и открываются в окне.

Свойства Window:

defaultStatus - Выводимое по умолчанию сообщение в строке состояния в нижней части окна броузера - [windowName].defaultStatus

status - Устанавливает текст основного или временного сообщения в строке состояния - [windowName].status.

frames - Содержит массив фреймов, содержащихся в окне - [windowName.] [parent.]frameName,[windowName.][parent.]frames[index]

name - Задает заголовок окна - windowRef.name

Методы:

window - Синоним текущего окна - [windowName.] window

alert - Выводит на экран диалоговое окно JavaScript Alert с кнопкой OK и определенным сообщением - [window].alert(AlertMessage)

open - Создает новый экземпляр окна - [window.]open("URL", "windowName" [,"windowFeatures"]);

close - Закрывает текущий экземпляр окна - [window.]close();

prompt - Отображает диалоговое окно ввода пользователя - [windowName.]prompt(message [inputDefault])

confirm - выводит на экран окна сообщения с кнопками Yes и No, и возвращает булевое значение true или false, в зависимости от нажатой кнопки - [window].confirm(ConfirmMessage)

setTimeout - Исполняет выражение по истечении указанного в миллисекундах промежутка времени - [window.]setTimeout(timerID)

Пример:

```
<SCRIPT LANGUAGE="JavaScript">
if (confirm("Уверены, что хотите просмотреть документ?") )
{
alert("Счастливого пути");
}
else
{
alert("Тогда оставайтесь");
window.close()
}
</SCRIPT>
```

Объект **document** - объект, создаваемый броузером во время загрузки страницы. Объект document является одним из основных в JavaScript и содержит информацию о текущем документе, такую как заголовок, цвет фона, имеющиеся в документе формы и т.п.. Эти свойства определяются в теге.

Свойства document:

action - Отображение атрибута ACTION тега <FORM> - document.formName.action; document.forms[index].action; action возвращает строку, состоящую из URL назначения для данных, введенных в форму. Это значение может быть установлено или изменено как до, так и после загрузки и форматирования документа.

alinkColor - Цвет активной гиперссылки - document.alinkColor

anchors - Массив всех якорей в документе - document.anchors(index)

bgColor - Фоновый цвет документа - document.bgColor

fgColor - Цвет текста, выводимого на странице - document.fgColor

forms - Массив объектов, соответствующих формам, созданных в тегах HTML в том же порядке - document.forms

lastModified - Строка только для чтения, хранящая дату последнего изменения текущего документа - document.lastModified

linkColor - Цвет гиперссылки в документе - document.linkColor

links - Массив гипертекстовых связей - document.links[index]

location - Возвращает строку с URL текущего документа - document.location

referrer - URL документа, который привел к текущему документу - document.referrer

title - Возвращает значение только для чтения, указанное внутри тега <TITLE> - document.title

vlinkColor - Возвращает или устанавливает цвет просмотренной гиперссылки - document.vlinkColor

images- Масив изображений, ссылки на которые заданы в текущем документе - document.images[index]

Методы:

open - Создает новый документ - document.open([MIMEtype]). Метод открывает поток вывода для методов write или writeln. Если тип MIME является версией text или image (например, text/html или image/gif), документ будет открыт для показа.

write - Записывает строку или несколько строк в окно документа - document.write(string)

writeln - Записывает строку или несколько строк в окно документа и добавляет символ новой строки в конце вывода - document.writeln(string)

close - Закрывает документ, посылает браузеру информацию об изменениях в документе - document.close()

clear - Очистка текущего документа - document.clear()

Пример:

```
<script language="JavaScript">
var bgc = document.bgColor;
var fgc = document.fgColor;
var url = document.location;
var lm = document.lastModified;
document.write("Цвет фона этой страницы <B>"+bgc+"</B>")
document.write("<BR>URL этой страницы <B>" +url+ "</B>")
document.write("<BR>Последние изменения внесены- <B>"+lm+ "</B>")
document.bgColor="00FFFF";
document.writeln("<b>Hello world!</b>");
</script>
```

Объект location сохраняет местоположение текущего документа в виде адреса этого документа.При управлении объектом location существует возможность изменять URL документа. Объект location связан с текущим объектом window - окном, в которое загружен документ.

Свойства объекта:

hash - Возвращает часть URL, начинающуюся с символа #, т.е. метку - document.linkName.hash; document.links[index].hash; document.location.hash

hostname - Возвращает или изменяет строку с именем домена или IP-адресом URL - location.hostname; linkName.hostname; links[index].hostname

href - Возвращает строку, содержащую полный URL текущего документа - location.href; linkName.href; links[index].href

pathname - Извлекает из URL ту его часть, которая содержит путь - location.pathname; link.pathname; links[index].pathname

Методов для объекта location не существует и с обработчиками событий он не связан

Пример:

```
<SCRIPT LANGUAGE="JavaScript">
document.write("Вы находитесь на следующей странице - " + window.location.href)
</SCRIPT>
<form>
<input type="button" value="Переход" onClick="location.href='forma/content.html '">
</form>
```

Примеры для выполнения:

Пример 1.

```
< html>
< body>
<A HREF="http://www.newmail.ru" onMouseOver="window.status='Бесплатный
хостинг'; return true">Ссылка</A>
<form>
<input type="text" size="30" onFocus="window.status='Текст в строке состояния';">
</ form>
< form>
< input type=" text" size="45" value="Впишите свое имя и щелкните по другой
строке" onBlur=" alert('Вы изменили ответ — уверены, что он правильный?');">
</ form>
< form>
< input TYPE=" text" size="45" value="Измените текст и щелкните по другой строке"
onChange=" window. status='Текст был изменен';">
</form>
<form>
<input TYPE="submit" onSubmit="parent.location='thanksalot.html'";>
</form></body>
</html>
```

Интересной возможностью Javascript является использование диалоговой панели ввода информации. Введенная в диалоговой панели текстовая строка выводится в окне браузера.

Пример 2

```
<HTML>
    <HEAD>
        <TITLE>Hello, world!</TITLE>
        <SCRIPT LANGUAGE="JavaScript">
        <!--
        function printHello()
        {
        szHelloStr=prompt("Введите приветственное сообщение:", "");
        document.write(szHelloStr);
        }
        // -->
        </SCRIPT>
    </HEAD>
    <BODY>
        <H1>JavaScript Test</H1>
        <SCRIPT LANGUAGE="JavaScript">
        <!--
        printHello();
        // -->
        </SCRIPT>
    </BODY>
</HTML>
```

Диалоговая панель ввода информации вызывается с помощью функции prompt. В качестве параметров функции передается вводное сообщение для пользователя и начальное значение запрашиваемой текстовой строки (в приведенном примере - пустое).

Пример 3.

```
<HTML>
    <HEAD>
        <TITLE>Hello, world!</TITLE>
    </HEAD>
    <BODY>
        <H1>JavaScript Test</H1>
        <A HREF="" onMouseover="alert('Hello, world!');">Select me!</a>
    </BODY>
</HTML>
```

Здесь для нас интересна строка оператора <A>. Напомним, что этот оператор обычно применяется для организации ссылок на другие документы HTML или файлы различных объектов. В данном случае поле ссылки параметра HREF пустое, однако

дополнительно в оператор <A> включена следующая конструкция: onMouseover="alert('Hello, world!');"

Она указывает, что при возникновении события onMouseover (наведение мыши) должна выполняться следующая строка программы JavaScript: alert('Hello, world!');

В настройках браузера может быть отключено выполнение сценария JavaScript. Выполнение скриптов включается следующим образом:

- в Internet Explorer: Сервис > Дополнительно > Разрешать запуск активного содержимого на моём компьютере;
- в Opera: Инструменты > Настройки > Дополнительно > Содержимое > Включить JavaScript;
- в Firefox: Инструменты > Настройки > Содержимое > Использовать JavaScript.

Окна и динамически создаваемые документы

Открытие новых окон в браузере является одной из преимущественных возможностей языка JavaScript. Вы можете либо загружать в новое окно новые документы (например, те же документы HTML), либо (динамически) создавать новые материалы. Посмотрим сначала, как можно открыть новое окно, загрузить в это окно HTML-страницу и, наконец, как его закрыть. open() и close() - это методы объекта window.

```
window.open('opened.html', 'joe', config='height=300,width=300')
self.name="main window"
```

Конфигурация нового окна

Информация об этом находится в круглых скобках . Вот схема, которой вы будете следовать:

('URL документа в новом окне', 'Название нового окна', config='параметры нового окна')

Есть множество подкоманд для команды config.

Вы имеете возможность управлять самим процессом создания окна. Рассмотрим основные свойства, которыми Вы можете управлять:

directories - Наличие в данном окне кнопок групп новостей
height - Высоту окна в пикселах
location - Наличие поля Location
menubar - Наличие меню
resizable - Наличие рамки окна, позволяющего менять его размеры
scrollbars - Наличие линеек прокрутки
status - Наличие строки состояния
toolbar - Наличие панели инструментов
width - Ширину окна в пикселах

По умолчанию атрибутам всегда присваивается значение yes, а размер нового окна (если он не задан) соответствует размеру предыдущего. Атрибуты можно указывать в произвольном порядке.

Дополнительная информация:

Тэги в новом окне

Всплывающее новое окно — это не просто рамка для страницы.

Открытие страницы в главном окне:

<A HREF="http://www.htmlgoodies.com" TARGET="main_window"></A>

У большого окна есть имя, main_window (главное). В скрипте это обозначено строкой *self.name="main_window"*. Добавляем в ссылку HREF TARGET="--" (цель) и указание на main_window.

А если надо, чтобы страница загружалась в маленьком окошке? Что ж, как оно называется? "joe". Просто нужно написать "joe" после команды target.

С помощью многократных команд window.open можно вызывать многократные окна. Только следите за тем, чтобы у каждого нового окна было свое имя. Можете

связывать окна ссылками при условии, что правильно указываете имена окон в команде target.

Закрыть окно

Вторая ссылка нового окна закрыла его. Вот как это сделано:

<A HREF="" onClick="self.close">Щелкните, чтобы закрыть</A>

Это обычная ссылка HREF, которая никуда не ведет. Видите пустые кавычки? Команда onClick="self.close" закрывает окно и никуда не ведет. self (само, себя) — это свойство может относиться к любому объекту. В нашем случае это свойство окна. Команда close (закрыть) закрывает окно.

Допустим, вы хотите открыть окно по команде, а не когда пользователь заходит на страницу. Вот как это можно сделать:

<A HREF="les11.htm" onClick="window.open('opened.html', 'joe', config='height=300,width=300')">Щелкните, чтобы открыть 'joe'</A>

Это ссылка HREF, которая направлена на самое себя. Главное окно остается открытым, так пускай перезагрузит самое себя. Команда onClick делает работу, а параметры содержатся в скобках().

Примечание: Чтобы страница не перезагружалась, можно поставить в кавычки после href знак # или вообще не указывать никакого адреса.

Пример 4*:*

```
<head>
<script language="JavaScript">
function openWin()
{
myWin= open("forma/close.htm", "displayWindow",
"width=400,height=300,status=no,toolbar=no,menubar=no");
}
</script>
</head>
<body>
<form>
<input type="button" value="Открыть новое окно" onClick="openWin()">
</form>
</body>
</html>
```

Динамическое создание документов

Рассмотрим такую замечательную возможность JavaScript , как динамическое создание документов. То есть Вы можете разрешить Вашему скрипту на языке JavaScript самому создавать новые HTML-страницы.

Пример 5:

```
<html>
<head>
<script language="javascript">
function newwin()
{
var OpenWindow=open("","newwindow", "height=300,width=300");
OpenWindow.document.write("<html>");
OpenWindow.document.write("<title>Новое окно</title>");
OpenWindow.document.write("<body bgcolor=white>");
OpenWindow.document.write("<a href='forma/content.htm' target=main>На главную</a>");
OpenWindow.document.write("<p><hr><p>")
OpenWindow.document.write("<a href='#' onClick='self.close()'>Закрыть окно</a>");
OpenWindow.document.write("</body></html>")
self.name="main";
}
</script>
</head>
<body>
<a href="#" onClick="newwin()">Открыть</a>
</body>
</html>
```

Объект Date

Объект содержит информацию о дате и времени. Этот объект имеет множество методов, предназначенных для получения такой информации. Кроме того объекты Date можно создавать и изменять, например путем сложения или вычитания значений дат получать новую дату. Для создания объекта Date применяется синтаксис:

dateObj = new Date(parameters)

где dateObj - переменная, в которую будет записан новый объект Date.

Аргумент parameters может принимать следующие значения: пустой параметр, например date() в данном случае дата и время - системные. строку, представляющую дату и время в виде: "месяц, день, год, время", например "March 1, 2000, 17:00:00" Время представлено в 24-часовом формате; значения года, месяца, дня, часа, минут, секунд. Например, строка "00,4,1,12,30,0" означает 1 апреля 2000 года, 12:30. целочисленные значения только для года, месяца и дня, например "00,5,1" означает 1 мая 2000 года, сразу после полночи, так, как значения времени равны нулю.

Данный объект имеет множество методов, свойств объект Date не имеет.

Методы Date.

getDate() Возвращает день месяца из объекта в пределах от 1 до 31

getDay() Возвращает день недели из объекта: 0 - вс, 1 - пн, 2 - вт, 3 - ср, 4 - чт, 5 - пт, 6 - сб.

getHours() Возвращает время из объекта в пределах от 0 до 23

getMinutes() Возвращает значение минут из объекта в пределах от 0 до 59

getMonth() Возвращает значение месяца из объекта в пределах от 0 до 11

getSeconds() Возвращает значение секунд из объекта в пределах от 0 до 59

getTime() Возвращает количество миллисекунд, прошедшее с 00:00:00 1 января 1970 года.

getTimeZoneoffset() Возвращает значение, соответствующее разности во времени (в минутах)

getYear() Возвращает значение года из объекта

setDate(day) С помощью данного метода устанавливается день месяца в объекте от 1 до 31

setHours(hours) С помощью данного метода устанавливается часы в объекте от 0 до 23

setMinutes(minutes) С помощью данного метода устанавливаются минуты в объекте от 0 до 59

setMonth(month) С помощью данного метода устанавливается месяц в объекте от 1 до 12

setSeconds(seconds) С помощью данного метода устанавливаются секунды в объекте от 0 до 59

setTime(timestring) С помощью данного метода устанавливается значение времени в объекте.

setYear(year) С помощью данного метода устанавливается год в объекте year должно быть больше 1900.

toGMTString() Преобразует дату в строковый объект в формате GMT.

toString() Преобразует содержимое объекта Date в строковый объект.

toLocaleString() Преобразует содержимое объекта Date в строку в соответствии с местным временем.

Date.UTC(year, month, day [,hours][,mins][,secs]) Возвращает количество миллисекунд в объекте Date, прошедших с с 00:00:00 1 января 1970 года по среднему гринвичскому времени.

Пример 6. Вывод текущей даты и времени.

```
<html>
<head>
<script language "JavaScript">
<--
function showh() {
var theDate = new Date();
document.writeln("<table cellpadding=5 width=100% border=0>" +
"<tr><td width=95% bgcolor=gray align=left>" +
"<font color=white>Date: " + theDate +
"</font></td></tr></table><p>");
}
showh();
//-->
</script>
</head>
</html>
```

Пример 7. Смена графических бэкграундов в зависимости от времени суток.

```
<html>
<script language "JavaScript">
<--
theTime = new Date();
theHour = theTime.getHours();
if (18 > theHour)
document.writeln("<body background='day.jpg' text='Black'>");
else
document.writeln("<body background='night.jpg' text='White'>");
//-->
</script>
</body>
</html>
```

Вероятно, вы успели заметить, что тег <body> создается в JavaScript-программе, а закрывается уже в статическом тексте HTML. Это вполне допустимо, так, как все теги

расположены в правильном порядке. В данном примере предполагается, что файлы рисунков находятся в том же каталоге. Вы можете здесь задать полный адрес URL.

Пример 8. Вывод текущей даты

Используется объект Date и метод write объекта document.

Исходник:

```
<SCRIPT LANGUAGE="JavaScript">
current_date = new Date();
document.write("<p><b>Текущая дата вашего компьютера:</b> " +
current_date + ".<p>");
</SCRIPT>
```

Результат: **Текущая дата вашего компьютера:** Sat Mar 17 00:04:03 UTC+0300 2013.

Пример 9.

Также можно вывести сокращенный вариант для этого используются методы getDate, getMonth, getYear. Обратите внимание, что нумерация месяцев начинается с нуля.

Исходник:

```
<SCRIPT LANGUAGE="JavaScript">
current_date = new Date();
document.write("<b>Число:</b> " + current_date.getDate() +
"." + current_date.getMonth() + "." + current_date.getYear() + ".");
</SCRIPT>
```

Результат: **Число:** 17.2.2007.

Пример 10.

Или для времени getHours, getMinutes, getSeconds такой вариант:

Исходник:

```
<SCRIPT LANGUAGE="JavaScript">
current_date = new Date();
document.write("<b>Время:</b> " + current_date.getHours() +
"." + current_date.getMinutes() + "." + current_date.getSeconds() + ".");
</SCRIPT>
```

Результат: **Время:** 0.4.3.

Пример 11. Идущие часы

Идущие часы можно поместить в строке статуса и в самом HTML документе.

Первый пример - скрипт, создающий часы в строке статуса при загрузке документа:

Создаем функцию

```
<script language="JavaScript">
function clock_status()
{
window.setTimeout("clock_status()",100);
today=new Date();
self.status=today.toString();
}
</script>
```

Вызываем функцию при загрузке документа в тэге <body>

```
<body onLoad="clock_status()">
```

Работающие часы смотрите в строке статуса.

Второй пример - обратите внимание, что функция вызывается в теле документа, а не в HTML-теге <body> как в предыдущем примере.

Создаем функцию

```
<script language="JavaScript">
function fulltime() {
var time=new Date();
document.clock.full.value=time.toLocaleString();
setTimeout('fulltime()',500)
}
</script>

Создаем форму
<form name=clock>
<input type=text size=17 name=full>
</form>

Вызываем функцию в теле документа
<script language="JavaScript">
fulltime();
</script>
```

Объект Math

Math - встроенный в JavaScript объект, дающий доступ к константам и математическим функциям.Объект Math делится на две части - свойства, содержащие константы и методы для реализации функций. Свойствами объекта Math являются математические константы:

E - Константа Эйлера. Приближенное значение 2.718

LN2 - Значение натурального логарифма числа два. Приближенное значение 0.693

LN10 - Значение натурального логарифма числа десять. Приближенное значение 2.302

LOG2_E - Логарифм e по основанию 2

LOG10_E - Десятичный логарифм e. Приближенное значение 0.434

PI - Число ПИ. Приближенное значение 3.1415

SQRT2 - Корень из двух

Методы объекта Math представляют собой математические функции:

abs() Возвращает абсолютное значение аргумента.

acos() Возвращает арккосинус аргумента

asin() Возвращает арксинус аргумента

atan() Возвращает арктангенс аргумента

ceil() Возвращает большее целое число аргумента, округление в большую сторону. Math.ceil(3.14) вернет 4

cos() Возвращает косинус аргумента

exp() Возвращает экспоненту аргумента

floor() Возвращает наибольшее целое число аргумента, отбрасывает десятичную часть

log() Возвращает натуральный логарифм аргумента

max() Возвращает больший из 2-х числовых аргументов. Math.max(3,5) вернет число 5

min() Возвращает меньший из 2-х числовых аргументов.

pow() Возвращает результат возведения в степень перврго аргумента вторым. Math.pow(5,3) вернет 125

random() Возвращает псевдослучайное число между нулем и единицей.

round() Округление аргумента до ближайшего целого числа.

sin() Возвращает синус аргумента

sqrt() Возвращает квадратный корень аргумента

tan() Возвращает тангенс аргумента

Объект String

Объект string представляет собой последовательность символов, ограниченная одинарными или двойными кавычками. Обычно присваивают какой-то переменной строку и используют ее как объект для вызова свойств или методов. Например, s="internet", а свойство s.**lenght** (длина строки) вернет значение 8. Рассмотрим некоторые методы объекта:

Большинство методов соответствует тегам HTML: **big()**, **fontcolor(arg), fontsize(arg)**, **small()**, **strike()**, **sub()**, **sup**()

anchor - Выводит строку на экран и делает ее якорем - textString.anchor(anchorName)

blink - форматирует строковый объект в виде мигающей строки - stringname.blink()

bold - Форматирует строковый объект жирным шрифтом - stringName.bold()

charAt - Возвращает символ, находящийся в заданной позиции строки - stringName.charAt(arg)

eval - Вычисляет строку как числовое выражение - eval(string)

fixed - Выводит строку на экран шрифтом фиксированной ширины - stringName.fixed()

italics - Отображает текст курсивом аналогично тегу <i> - stringName.italic()

indexOf - Возвращает позицию символа или подстроки в строке, начиная поиск сначала - stringName.indexOf()

lastIndexOf - Возвращает позицию символа или подстроки в строке, начиная поиск с конца - stringName.lastIndexOf()

link - Создает новую гиперссылку на другой URL - stringName.link(argument)

substring - Позволяет извлечь подстроку длиной arg2, начиная с позиции arg1 - stringName.substring(arg1, arg2)

toLowerCase - Преобразует все символы строки к нижнему регистру - stringName.toLowerCase()

toUpperCase - Преобразует все символы строки к верхнему регистру - stringName.toUpperCase()

Выявление и исправление ошибок

Чем больше опыт вашей работы в JavaScript, тем чаще в ваших сценариях возникают ошибки. Они вызывают раздражение, их трудно найти, но скоро вы научитесь отыскивать их и исправлять.

Ошибки (bugs) — это проблемы, возникающие в сценарии. Процесс отыскания и исправления ошибок называется отладкой (debugging). Слово bug в английском языке означает «насекомое» или «жучок».

Самыми распространенными являются следующие:

- синтаксические ошибки;
- ошибки времени выполнения;
- логические ошибки;
- ошибки приоритета операций.

Синтаксические ошибки - наиболее распространенный тип ошибок, знакомый каждому программисту. Синтаксис - это набор грамматических правил языка программирования, а поскольку компьютеры более требовательны к языку, чем люди от вас требуется чрезвычайная внимательность.

Сценарий JavaScript проверяется на наличие синтаксических ошибок во время загрузки в браузер. Таким образом, подобные ошибки выявляются сразу.

Ошибки времени выполнения возникают в тех случаях, когда в коде сценария JavaScript поставлена невыполнимая задача. Они называются так потому, что могут возникнуть в любой момент выполнения сценария.

Логические ошибки - это не столько ошибки языка JavaScript, сколько самого сценария. Вы можете написать сценарий для расчета налога с продаж, но вместо того

чтобы прибавлять налог к сумме покупки, программа будет его вычитать. Это логическая ошибка.

Ошибки приоритета операций — по сути, те же логические ошибки, но они связаны с порядком выполнения математических операций.

Сообщения об ошибках

Если браузер встретит ошибку в коде JavaScript, то весь сценарий не будет исполнен. Когда скрипт большой и много строк кода, то найти ошибку бывает сложно. Отыскать её помогает сообщение об ошибке, которое может выдавать браузер. Но для этого его нужно настроить следующим образом.

- в Internet Explorer: Сервис > Дополнительно > Обзор > Показывать уведомление о каждой ошибке сценария;
- в Opera: Инструменты > Настройки > Дополнительно > Содержимое > Настроить JavaScript > Открывать консоль при ошибке;
- в Firefox: Инструменты > Консоль ошибок.

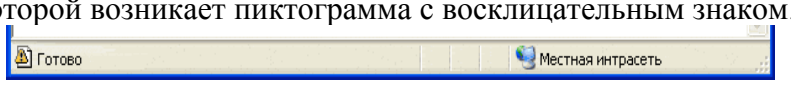
Как найти ошибку

Пример:

```
<script language="JavaScript">
<!-- Маскируемся!
alrtt("Не работает!");
// Снимаем маскировку. -->
</script>
```

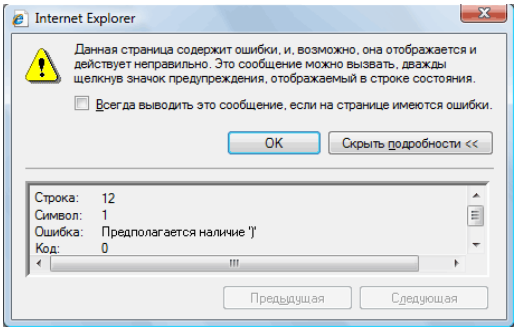
Здесь допущена простая синтаксическая ошибка - опечатка в слове alert.

На рисунке показана ошибка применительно к браузеру Internet Explorer. При загрузке страницы в строке сообщений мелькает надпись Done, but with errors on the page (Выполнено, но с ошибками на странице). Затем эта фраза сменяется надписью Done (Готово), рядом с которой возникает пиктограмма с восклицательным знаком.



Пиктограмма, предупреждающая об ошибке, в строке состояния браузера Internet Explorer

Двойной щелчок по этой пиктограмме вызывает диалоговое окно с информацией об ошибке



Диалоговое окно с информацией об ошибке

Обнаружение ошибок

Что-то начинает проясняться.

- вам придется открыть файл и сосчитать строки, пока не доберетесь до нужной;
- сообщенная информация не совсем понятна;
- эта информация не всегда бывает точной.

Однако пользователи Internet Explorer могут упростить поиск ошибок, применив специальную программу, разработанную компанией Microsoft. Она называется Microsoft Script Debugger.

Рекомендация Программа Microsoft Script Debugger распространяется бесплатно. Загрузите ее с сайта http://www.Microsoft.com/scripting.

Указания по выполнению

Задание 1. Измените **Пример 3** так, чтобы диалоговая панель возникала не при наведении курсора мыши, а при выборе ссылки (событие onClick).

Задание 2. Измените **Пример 3** так, чтобы при наведении курсора мыши на ссылку, выполнялась бы процедура, выводящая в окно браузера фразу "Hello, word!".

Задание 3. Напишите скрипт, который сначала выводит на экран диалоговое окно, а затем, после нажатия кнопки "ОК", в окне браузера пишет фразу "Hello, world!".

Задание 4. Напишите скрипт, который запрашивает у пользователя информацию, а затем выводит ее в диалоговом окне.

Задание 5. Составьте документ так, чтобы диалоговое окно для ввода информации предлагалось только после наведения курсора мыши на ссылку, и введенная пользователем текстовая строка выводилась бы в виде диалогового окна, или в окно браузера.

Задание 6. Написать скрипт, реализующий сложение двух чисел по щелчку на кнопке (Используйте функцию eval)

Задание 7. Создать форму, которая будет иметь три элемента: поле ввода с просьбой ввести имя, при вводе которого в строке состояния должны появиться слова: «Впишите сюда свое имя»; два поля для флажков с вопросом о том, что больше нравится пользователю, мороженое или шоколад, которые должны отослать в строку состояния слова: «Вы выбрали...» и выбор пользователя; кнопку отправки данных, при нажатии на которую должно выскочить окно предупреждения, благодарящее пользователя за участие в опросе.(см. Пример1)

Задание 8. Создать документ, в котором располагалась бы кнопка, при нажатии выводящая диалоговое окно, которое открывает второй документ при нажатии на кнопку "Ok" и закрывает данный документ при нажатии на "Cancel" (метод **confirm( )**).

Задание 9. Написать функцию, которая открыла бы окно с зеленым фоном и приветствием: «Привет, имя пользователя, вот твое окно!» Создайте кнопку, которая закроет это окно. За основу возьмите Пример 4-5.

Задание 10. Напишите скрипт, который откроет новое окно со всеми характеристиками. Пусть оно будет размером 250 на 300 пикселей и с двумя ссылками:

- Одна откроет новую страницу в главном окне.
- Вторая откроет новую страницу в том же окне.

Страница, которая откроется в том же маленьком окне, должна содержать ссылку, закрывающую окно. За основу возьмите Пример 4-5.

Задание 11. Написать скрипт, содержащий текущее время и дату в текстовых полях время и дата

Задание 12. Сделайте на первой странице идущие часы (чч.мм.сс)

Задание 13. Найти максимальный элемент массива из 5 элементов. Результат вывести по щелчку на кнопке. Использовать методы объекта Math.

Задание 14. Задан текст. Определить позицию первого символа "a" в тексте и вывести на экран строку, начиная с этого символа, длиной 3 символа. Использовать методы объекта String.

Содержание отчета

1. Цель
2. Создание сценариев JavaScript.
3. Обращение к свойствам и методам объектов.
4. Иерархия объектов JavaScript.
5. События JavaScript.
6. Выводы

Контрольные вопросы

1. Иерархия объектов JavaScript.
2. Назначение объекта window.

3. Перечислите свойства и методы объекта window, которые использовали при выполнении лабораторной работы.
4. Назначение объекта document.
5. Перечислите свойства и методы объекта document, которые использовали при выполнении лабораторной работы
6. Для чего предназначено свойство *href* объекта *location?*
7. Для чего используется функция eval?
8. Опишите синтаксис конфигурации нового окна.
9. Перечислите возможные параметры нового окна.
10. В каких случаях используют после *href* знак *#?*
11. Какой метод используется для закрытия окна?
12. Что означает запись: *self.name="main_window"?*
13. Объясните принцип динамического создания документов.
14. Назначение объекта *Date*.
15. Перечислите методы объекта Date, которые вы использовали при выполнении лабораторной работы.
16. Как можно обратиться к текстовому полю формы, с целью отображения в нем данных, например, даты*?*
17. Назначение объекта Math.
18. Назначение объекта String.
19. Перечислите методы объектов Math и String, которые вы использовали при выполнении лабораторной работы.

Лабораторная работа №18
Подготовка и оптимизация графики на веб-странице. Создание баннера

Цель: получить практические навыки создания баннера**.**

Задача 1. Написать код ротатора баннеров на JavaScript.

Ротатор баннеров — это инструмент случайного показа ссылок и баннеров на страницах сайта. Ротатор полезен тем, что позволяет транслировать баннеры по очереди и при этом не занимает много пространства на странице. Также по баннерам ведется статистика показов и кликов, опираясь на которую можно гибко управлять их отображением.

HTML:

```
<a href="#" id="adLink" title="" target="_blank">
<img src="images/1.jpg" id="adBanner" width="2000" height="250" alt="" /></a>
<script>
    let imgs = new Array("images/1.jpg", "images/2.jpg", "images/3.jpg");
    let links = new Array("#", "#", "#");
    let alt = new Array("баннер-1", "баннер-2", "баннер-3");
    let title = new Array("title-1","title-2","title-3");
    let currentAd = 0;
    let imgCt = 3;
    function cycle() {
    if (currentAd == imgCt) {
    currentAd = 0;
    }
    let banner = document.getElementById('adBanner');
    let link = document.getElementById('adLink');
    banner.src = imgs[currentAd]
    banner.alt = alt[currentAd]
    banner.title = title[currentAd]
    document.getElementById('adLink').href = links[currentAd]
    currentAd++;
}
window.setInterval("cycle()", 3000);
```

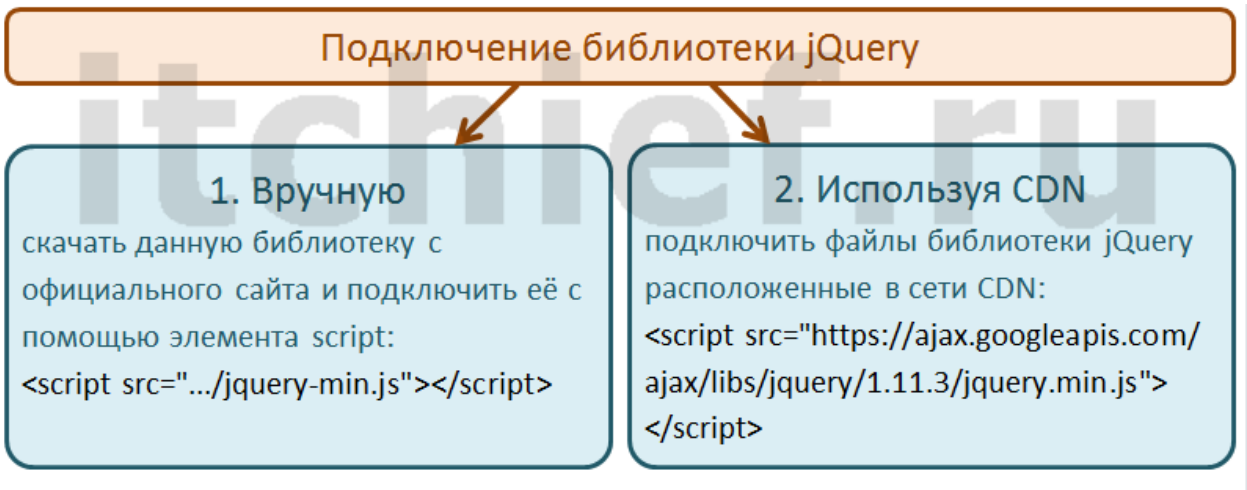
JavaScript:

```
<a href="#" id="adLink" title="" target="_blank">
<img src="images/1.jpg" id="adBanner" width="2000" height="250" alt="" /></a>
```

Теоретическая часть

jQuery — это библиотека, написанная на языке JavaScript, которая предназначена для кроссплатформенного манипулирования HTML страницей

после того, как она отобразится в браузере. Кроме этого, jQuery содержит инструменты помогающие перехватывать и обрабатывать события, создавать анимацию на веб-странице и методы, которые позволяют взаимодействовать с сервером без перезагрузки страницы (AJAX).



Как использовать библиотеку jQuery?

После подключения библиотеки jQuery к HTML документу её можно использовать с помощью функции jQuery (window.jQuery()). Данную функцию можно использовать не только по имени jQuery, но и по более короткому и красивому псевдониму - знаку $. Не забываем, что в JavaScript функции являются тоже объектами, следовательно, $ - является объектом.

```
window.jQuery = window.$ = jQuery = $;
```

Работа с jQuery всегда начинается с функции jQuery() или её псевдонима $(). Данная функция может принимать один или два аргумента. В зависимости от переданных ей значений аргументов функция jQuery() выполняет то или иное действие. В большинстве случаев данную функцию используют для выбора элементов, создания элементов или вызова анонимной функции. Если, например Вы будете использовать функцию jQuery для выбора элементов, то получите объект jQuery, состоящий из выбранных элементов. Следующий этап при работе с библиотекой jQuery обычно сводится к вызовам различных методов jQuery.

Задача 2. Написать код ротатора баннеров на jQuery.

HTML:

```
<div id="rotator">
    <ul>
     <li class="show"><a href="#"><img src="images/1.jpg" width="300" height="250"  alt="pic1" /></a></li>
     <li><a href="#"><img src="images/2.jpg" width="300" height="250"  alt="pic2" /></a></li>
     <li><a href="#"><img src="images/3.jpg" width="300" height="250"  alt="pic3" /></a></li>
    </ul>
   </div>
```

CSS:

```css
<style>
    div#rotator {position:relative; height:150px; margin-left: 15px;}
    div#rotator ul li {float:left; position:absolute; list-style: none;}
    div#rotator ul li.show {z-index:500;}
</style>
```

jQuery:

```
<script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
<script>
function theRotator() {
 // Устанавливаем прозрачность всех картинок в 0
 $('div#rotator ul li').css({opacity: 0.0});
 // Берем первую картинку и показываем ее (по пути включаем полную видимость)
 $('div#rotator ul li:first').css({opacity: 1.0});
 // Вызываем функцию rotate для запуска слайдшоу, 5000 = смена картинок происходит раз в 5 секунд
 setInterval('rotate()', 5000);
}

function rotate() {
 // Берем первую картинку
 let current = ($('div#rotator ul li.show')?  $('div#rotator ul li.show') : $('div#rotator ul li:first'));
 // Берем следующую картинку, когда дойдем до последней начинаем с начала
 let next = ((current.next().length) ? ((current.next().hasClass('show')) ?
 $('div#rotator ul li:first') :current.next()) : $('div#rotator ul li:first'));
 // Расскомментируйте, чтобы показвать картинки в случайном порядке
 // let sibs = current.siblings();
 // let rndNum = Math.floor(Math.random() * sibs.length );
 // let next = $( sibs[ rndNum ] );
 // Подключаем эффект растворения/затухания для показа картинок, css-класс show имеет больший z-index
 next.css({opacity: 0.0})
 .addClass('show')
 .animate({opacity: 1.0}, 1000);
 // Прячем текущую картинку
 current.animate({opacity: 0.0}, 1000)
 .removeClass('show');
};

$(document).ready(function() {
// Запускаем слайдшоу
theRotator();
});
</script>
```

Лабораторная работа № 19

Словарь схемы сайта. Логическая схема сайта

Цель: *получить практические навыки проектирования логической схемы сайта, создания словаря схемы сайта.*

Ход работы:

1. Изучить теоретический материал.
2. Выполнить задания в соответствии с указаниями.
3. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
4. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Грамотно продуманная организация всех разделов, подразделов, категорий и просто страниц сайта крайне важна для успеха его продвижения. Если пользователю будет сложно ориентироваться на сайте, он просто закроет его, так и не совершив целевого действия.

Для чего нужна структура сайта

Следующий этап после подбора и анализа ключевых слов – разработка логической структуры сайта и разбивка ключевых слов по посадочным страницам.

Чтобы понять, как это выглядит, лучше всего воспользоваться инструментом наподобие Cacoo.

При разработке логической структуры сайта нужно учитывать доступность важных разделов для пользователей. Посетители не должны искать необходимые разделы, они (разделы) должны быть всегда на виду. Пользователь не должен совершать более двух-трёх кликов мышкой, чтобы попасть в нужный раздел. Кроме того, уровень вложенности не глубже третьего упростит индексацию сайта поисковыми системами.

У крупных интернет-порталов и магазинов может быть довольно сложная разветвлённая структура сайта, состоящая из разделов, категорий, подкатегорий и т. д. При её разработке важно учитывать те же общие принципы, что и для любого другого сайта.

Семантическая схема сайта

Когда общая концепция сайта готова, известны основные разделы и категории, необходимо распределить семантическое ядро по страницам сайта (так называемые “посадочные страницы”). Если семантического ядра ещё нет, можно воспользоваться несколькими способами:

- Осуществить ручной подбор запросов в Яндекс Вордстате;
- Воспользоваться программой KeyKollector.

В программе KeyKollector есть возможность создать древовидную структуру проекта. Удобно распределять семантическое ядро по разным папкам проекта внутри программы. По завершению подбора ключевых слов и распределения их по папкам получится практически готовая логическая структура сайта с посадочными страницами, разделами, подразделами и категориями.

Выводы

- Разработка структуры сайта – важный этап, который необходимо прорабатывать до начала работ по созданию и SEO-продвижению сайта;
- Структура напрямую влияет на то, какую CMS выбрать для сайта;
- Для создания структуры сайта необходимо иметь на руках готовое семантическое ядро;
- Опыт конкурентов играет большую роль при создании удачной структуры сайта

Структура сайта в виде схемы и требования предъявляются к правильно разработанному проекту. Главным образом их формируют поисковики, поэтому под них

обычно подстраиваются вебмастера. Однако не стоит забывать, что первым делом необходимо позаботиться о посетителях, а уж затем о роботах. Поисковые системы анализируют сайт по-своему, беря во внимание URL. Надо сказать, что структура сайта и URL являются одинаковыми понятиями. У каждого поисковика свои требования: Требования от Яндекс:

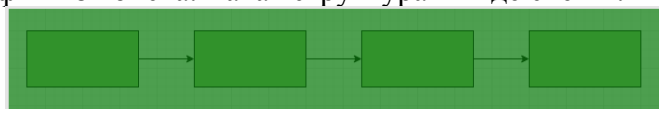
- Старайтесь поддерживать четкое расположение ссылок.
- Все документы должны относиться к определенному разделу.
- Кроме того, на каждую страницу должна идти хотя бы одна ссылка, много ресурсов образовательного учреждения не берут во внимание.
- Не забывайте про карту проекта, она может ускорить индексацию.
- Один адрес должен быть доступным только по одной ссылке.

Требования от Google:

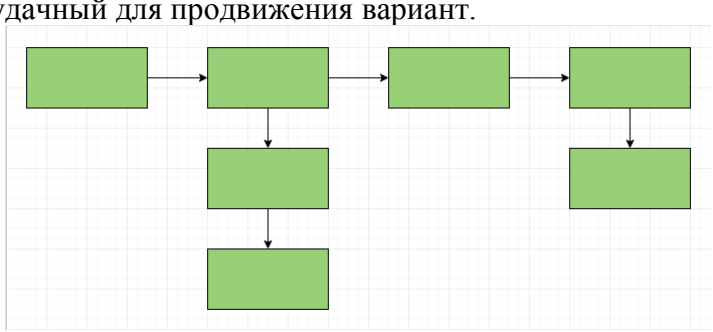
- Структуру следует делать простой.
- URL понятны для обычных пользователей.
- Применяйте слова, а не идентификаторы. Не стоит делать очень длинные и сложные адреса.

Виды структуры сайта

Линейная структура. Элементарная логика – каждая страница ссылается на другую страницу и на главную. Такую структуру хорошо применять на сайтах-презентациях, портфолио и других специфических продуктах, которые преследуют цель ознакомить посетителя со всеми страницами в определенной последовательности. Вес страниц здесь перетекает от главной к последней странице через все остальные. Успешно продвигать можно только главную, поэтому для привлечения посетителей эта структура не очень подходит. Оставим ее тем, кто использует сайты для своих целей и не собирается получать трафик из поиска. Такая структура в виде схемы:



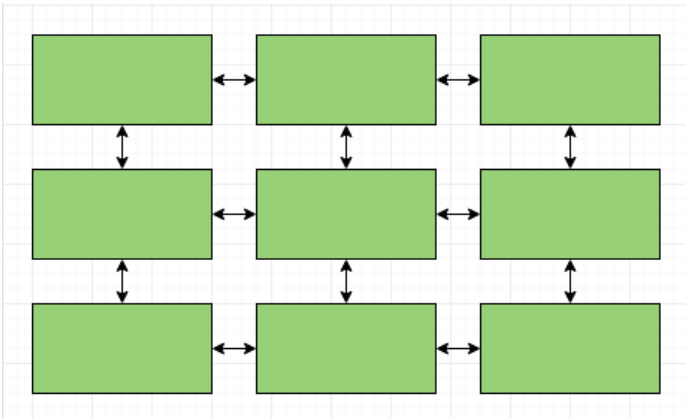
Линейная с ответвлениями. Принцип тот же, но здесь вы можете использовать несколько продуктов на одном сайте, с которыми будете знакомить так же постранично. Например, это может быть онлайн-библиотека какого-то автора с несколькими книгами. Здесь вес опять же передается от главной к последней странице, правда таких страниц уже несколько. И снова неудачный для продвижения вариант.



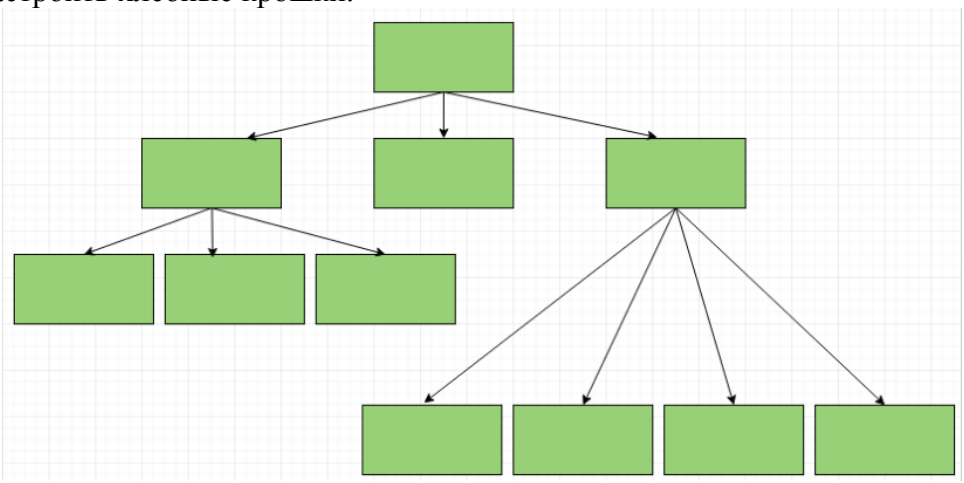
Мы увеличиваем посещаемость и позиции в выдаче. Вы получаете продажи и платите только за реальный результат, только за целевые переходы из поисковых систем

Заказывайте честное и прозрачное продвижение

Блочная структура. Здесь все страницы ссылаются на несколько других, которые равнозначны между собой. Такую структуру неплохо применять для конкретного продукта, когда каждую страницу можно использовать, как описание какого-то отдельного свойства/достоинства и их совокупностей. С распределением веса здесь все вполне неплохо, страницы уже перелинкованы и отдают свой вес на главную, что позволяет продвигать ее более эффективно. Но такая структура весьма специфична и применять ее можно далеко не везде. Пример структуры сайта:



Древовидная структура. Именно древовидная структура является наиболее универсальным вариантом и именно ее вы будете применять в 99% случаев. Смысл в том, что для каждого направления у вас будет своя ветка, для каждой услуги или товара у вас будет отдельное ответвление. То есть, те самые привычные нам разделы и подразделы. Эта структура позволяет передавать дополнительный вес как на главную, так и на разделы (каждая страница раздела будет ссылаться не только на главную, но и на свой раздел, достаточно настроить хлебные крошки.



В URL:
site.ru/divani/
site.ru/divani/tkani.html
site.ru/divani/kozha.html
site.ru/pyfiki/
site.ru/stylya/
site.ru/stylya/derevo.html
site.ru/stylya/plastic.html
site.ru/stylya/rotang.html
site.ru/stylya/metall.html

Именно древовидная структура сайта более эффективна. Поэтому она привлекательна и интересна и с точки зрения продвижения, и с точки зрения удобства.

Как создать структуру своего сайта. Виды сайтов и их структура

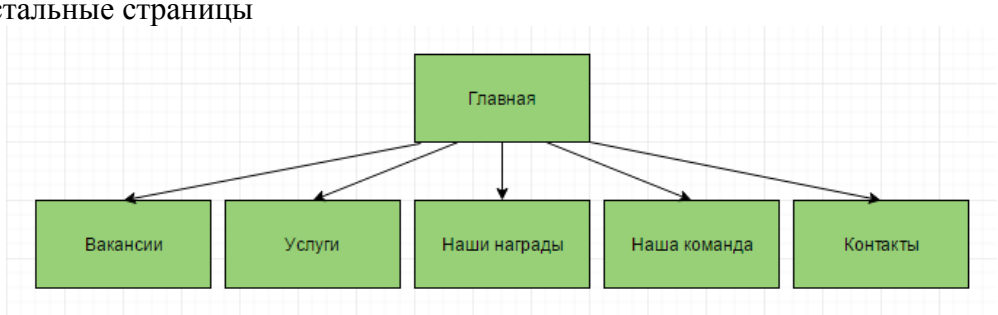
После того, как вы определились с видом структуры, который будете использовать., нужно понять, какой она должна быть именно у вашего сайта: как будут организованы страницы, разделы, подразделы и прочее. Здесь все зависит от вида и назначения сайта. Ниже представлены именно логические структуры сайтов. Расположение блоков на видимой части сайта – той, что вы видите на экране монитора (кнопочки, меню и т.д.), – может быть любым, главное, чтобы оно казалось вам правильным.

Визитка

Стандартная структура сайта-визитки обычно проста и состоит всего из двух уровней:

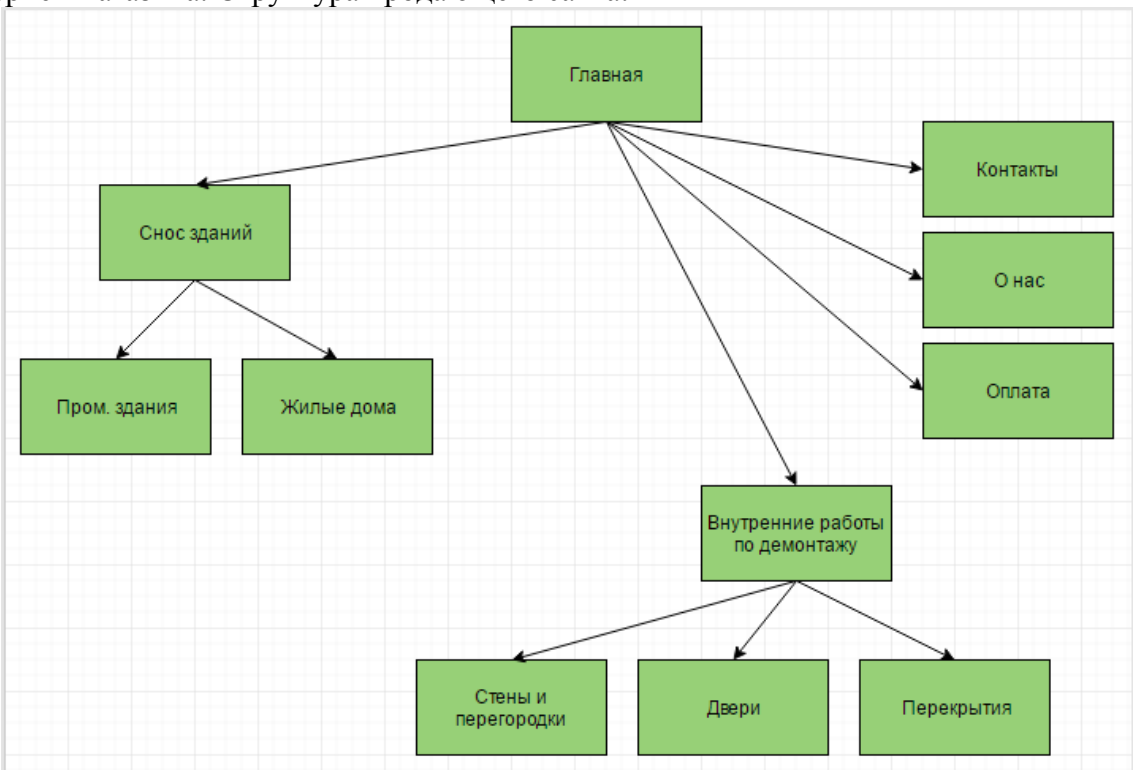
главная

остальные страницы



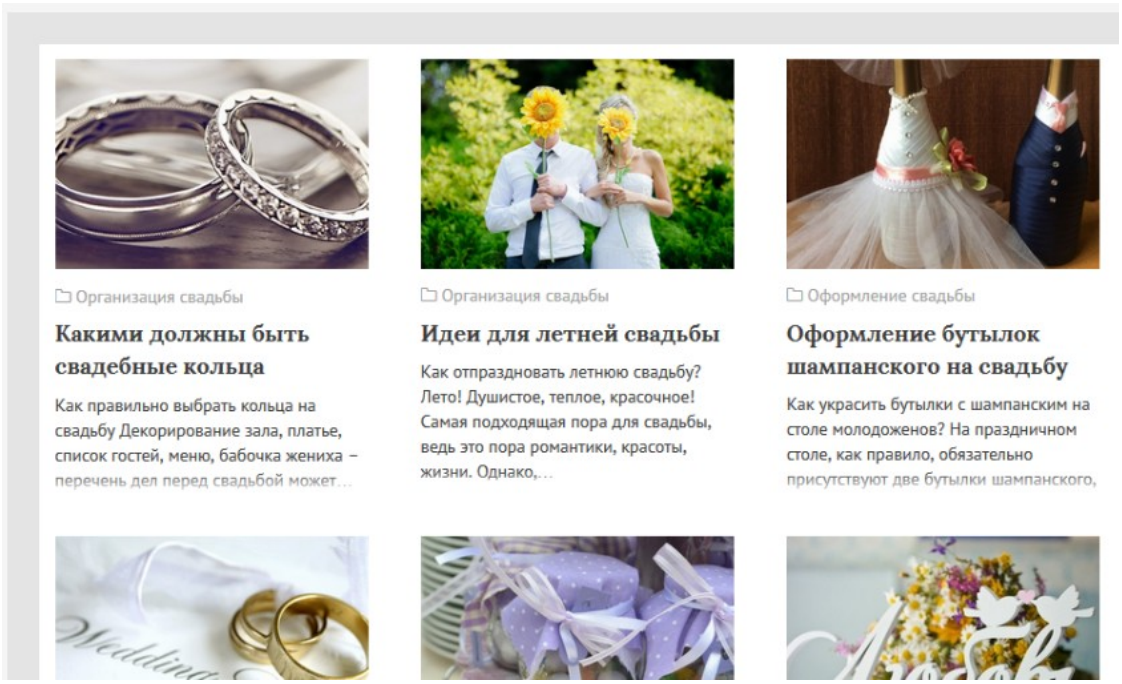
Коммерческий сайт

Здесь уже сложнее – нужно будет внедрять дополнительные уровни страниц. Это будет понятно из семантического ядра. Основными страницами будут главные направления вашей деятельности, страницами второго уровня – их разновидности. Такое решение идеально для сайтов услуг и любых других коммерческих сайтов без функций интернет-магазина. Структура продающего сайта:

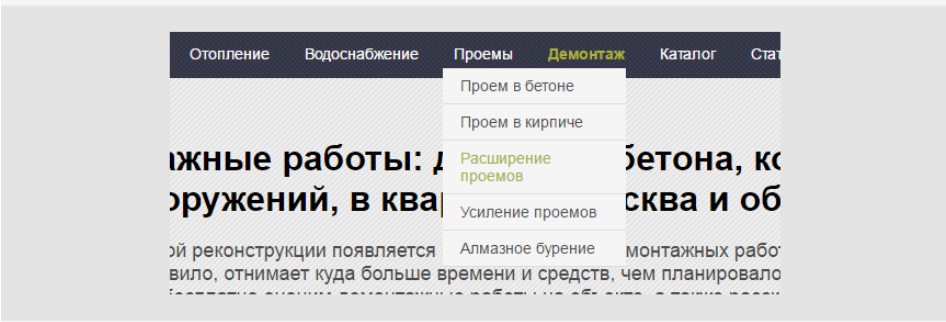


Информационный сайт и блог

Структура информационного сайта похожа на структуру коммерческого, с той лишь разницей, что уровни будут представлены разделами (именно разделами, а не страницами 1-го уровня) и страницами. Если представить это на сайте, то страницы раздела – это страница, содержащая много ссылок на страницы ему принадлежащие (листинг). Сама по себе страница раздела может не нести никакой пользы в плане дополнительной информации. Пример:

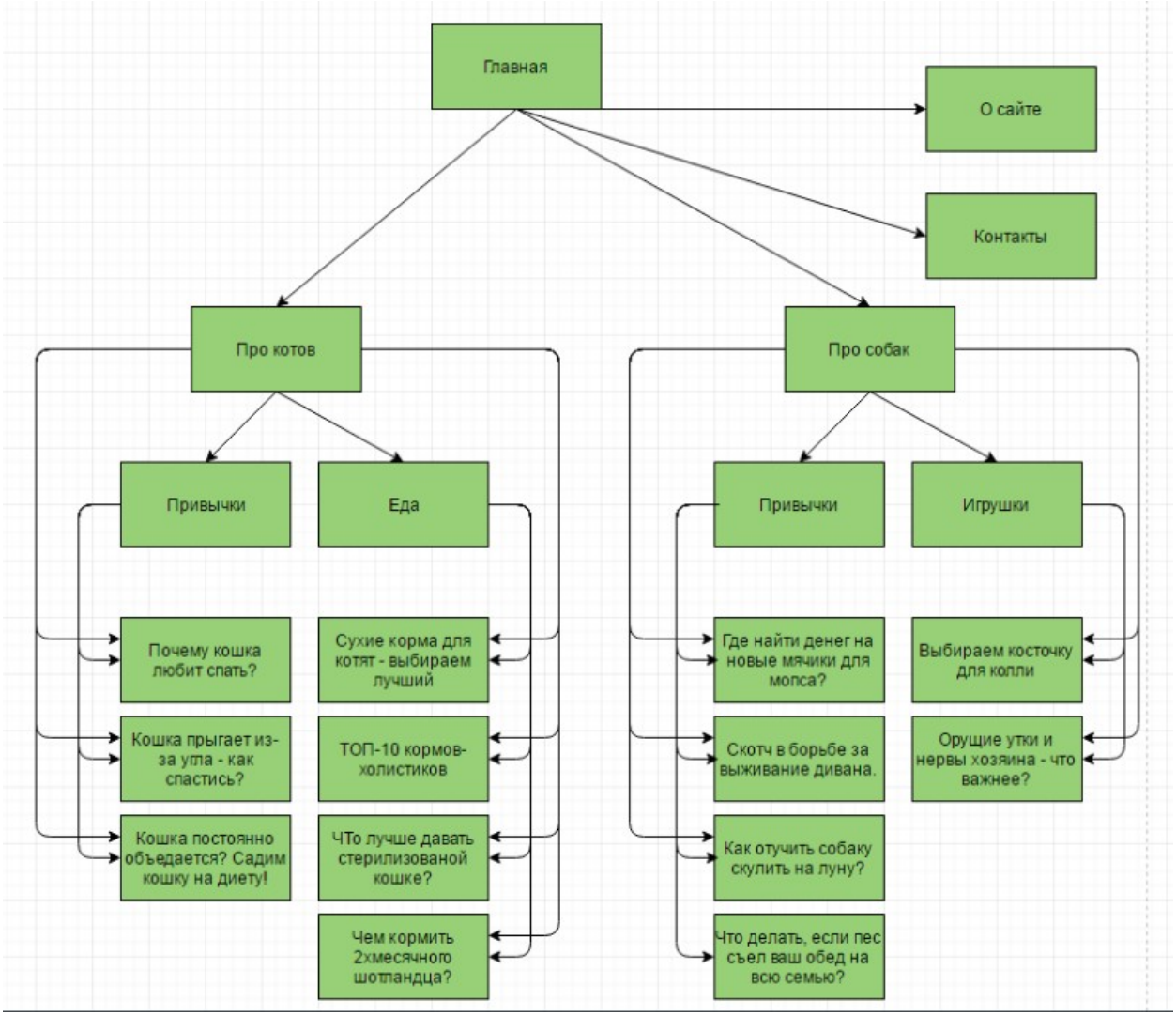


На страницах коммерческих сайтов такого обычно нет (попасть на подстраницу можно из меню) и сама по себе страница является важной информационной единицей. Пример сайта услуг:



На информационных сайтах страница со статьей доступна и из подраздела, и из раздела, и, в некоторых случаях, с главной.

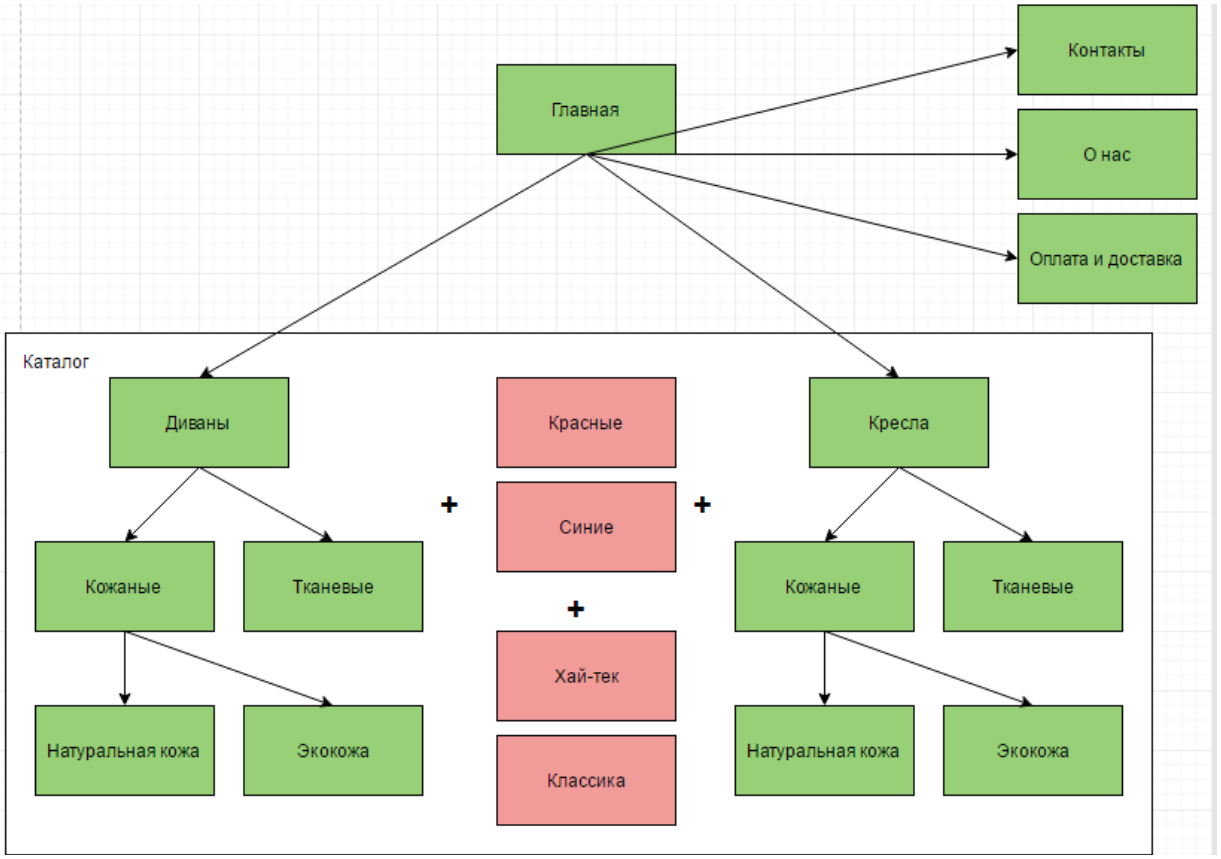
Определиться с тем, какие у вас будут разделы и подразделы, вы сможете и при проработке концепции сайта (когда будете продумывать, о чем будете писать в личном блоге и как читателю будет удобнее находить нужное на сайте), и при оценке СЯ (для тех случаев, когда сайт создается под семантическое ядро для заработка на рекламе). Структура информационного сайта на схеме:



Интернет-магазин

Это уже совершенно иной уровень организации сайта. Чтобы охватить все возможные запросы, которые могут ввести в поисковой строке ваши потенциальные покупатели, вам нужно будет внедрить не только систему разделов и подразделов, но и добавить на сайт фильтры. Главное, правильно определить, что пойдет в фильтры, а что будет основным свойством продаваемого товара. Так, например, для магазина диванов логично будет разделить диваны на кожаные и тканевые (материал), а не на синие и красные (цвет). Хотя, если ваша фишка – это разноцветные диваны, то будет лучше разделить их по цветам, сделав материал второстепенным признаком и вынести его в фильтр. Фильтры нужны тогда, когда одному товару присуще сразу несколько качеств, которые интересуют людей в запросах, например, «цвет + форма» (кровать белая круглая) или «размер + форма + материал» (большой угловой кожаный диван).

Если вы понимаете, что не можете собственноручно создать страницы со всевозможными вариантами сочетаний параметров товаров – вам нужны фильтры. Если у ваших товаров всего один-два параметра (например, если вы продаете конкретный вид продукции - бамбуковые одеяла, и единственный параметр, который изменяется, это размер), можно обойтись без фильтров. Фильтры могут быть одинаковыми для каждого уровня и раздела каталога, могут разрабатываться отдельно – все зависит от специфики. Упрощенная структура сайта интернет-магазина:



Практическая часть

Задание. Разработайте логическую структуру сайта в соответствии с индивидуальным вариантом, основываясь на нижеприведенном примере.

Используйте доступную программу для создания структуры сайта (и не только) draw.io, в которой и сделаны схемы, приведенные в данных указаниях.

Пример проектирования структуры сайта. Мастер-класс для коммерческого семантического ядра

Чтобы пришло понимание, как выделять разные уровни страниц, рассмотрим, как это делается на примере конкретного семантического ядра. Слова отобраны только для примера, поэтому пытаться найти там признаки идеального СЯ нет смысла. Мы будем составлять правильную логическую структуру сайта, это структура, в которой уровни страниц определяются с помощью простой человеческой логики.

Сейчас много всяких программ, которые помогают в кластеризации, но для того, чтобы понять, как вообще ведется разработка структуры сайта, нужно сперва научиться делать все своими ручками и мозгами, а уже потом доверяться программам. Поэтому и показываю «на пальцах». В общем, в работе такой вот список слов:

беременна фотосессия	фотограф воронеж цены
бесплатные фотосессии в воронеж	фотограф мужчин
выписка из роддома фотосессия	фотограф на выписку
детская фотосессия воронеж	фотограф на свадьбу
детские семейный фотограф	фотограф на свадьбу воронеж
детские фотографы воронежа	фотограф на свадьбу недорого
детский фотограф воронеж	фотограф на свадьбу цены
ищу фотографа воронеж	фотограф новорожденного
недорогие фотографы воронежа	фотограф новорожденных
недорогие фотосессии воронеж	фотограф свадебная фотосъемка
подарочный сертификат на фотосе	фотограф свадьбу день
профессиональная фотосессия	фотограф студия
профессиональные фотографы вор	фотограф цены
профессиональный фотограф	фотографы воронежа
профессиональный фотограф фото	фотографы воронежа отзывы
профессиональный фотограф цена	фотографы воронежа сайт
свадебная фотосессия	фотосессии в воронеже недорого
свадебная фотосессия воронеж	фотосессия беременности
свадебный фотограф	фотосессия беременных
свадебный фотограф воронеж	фотосессия в воронеже в студии цена
свадебный фотограф фотосессия	фотосессия в студии
свадебный фотограф цены	фотосессия в студии цена
свадьба фотосессия	фотосессия воронеж
семейная фотосессия	фотосессия воронеж в студии
семейная фотосессия в студии	фотосессия воронеж цены
семейная фотосессия воронеж	фотосессия воронеже бесплатно
семейная фотосессия летом	фотосессия выписка
семейная фотосессия на природе	фотосессия детей воронеж
семейная фотосессия на природе	фотосессия для беременных воронеж
семейная фотосессия на природе	фотосессия для мужчин
семейные фотосессии с детьми	фотосессия мама с дочкой
семейный фотограф	фотосессия новорожденного
семейный фотограф воронеж	фотосессия новорожденных
сертификат на фотосессию	фотосессия с дочкой
услуги профессионального фотогра	фотосессия с мужем
услуги фотографа воронеже	фотосессия с мужем на природе
услуги фотографа цены	фотосессия свадьбы воронеж
фотограф воронеж недорого	фотосессия цена

Теперь нужно продумать, какие группы можно выделить. Чтобы это сделать, включаем логику. Разнесем по отдельным группам те слова, которые имеют одну суть. Например, отделим беременных от детей, недорогие фотосессии от бесплатных, фотосессию в студии от фотоссесии на выписке и т.д. Вот что получится:

недорогие фотографы воронежа	фотосессия в воронеже в студии цена	семейная фотосессия воронеж
недорогие фотосессии воронеж	фотосессия в студии	семейная фотосессия летом
фотограф воронеж недорого	фотосессия в студии цена	семейный фотограф
фотосессии в воронеже недорого	фотограф студия	семейная фотосессия
	фотосессия воронеж в студии	семейный фотограф воронеж
бесплатные фотосессии в воронеже		семейная фотосессия на природе
фотосессия воронеже бесплатно	фотосессия для мужчин	семейная фотосессия на природе с ребенком
	фотограф мужчин	семейная фотосессия на природе летом
профессиональная фотосессия		семейная фотосессия в студии
профессиональные фотографы воронежа		
услуги профессионального фотографа	свадебная фотосессия	фотосессия с дочкой
профессиональный фотограф	свадебная фотосессия воронеж	фотосессия мама с дочкой
профессиональный фотограф фотосессии	свадебный фотограф	семейные фотосессии с детьми
услуги фотографа воронеже	свадебный фотограф фотосессия	
ищу фотографа воронеж	свадебный фотограф цены	фотосессия для беременных воронеж
фотографы воронежа	свадьба фотосессия	фотосессия беременности
фотосессия воронеж	свадебный фотограф воронеж	фотосессия беременных
фотографы воронежа сайт	фотосессия свадьбы воронеж	беременна фотосессия
	фотограф на свадьбу	
профессиональный фотограф цена	фотограф на свадьбу воронеж	фотосессия с мужем
фотосессия цена	фотограф на свадьбу недорого	фотосессия с мужем на природе
услуги фотографа цены	фотограф на свадьбу цены	
фотограф воронеж цены	фотограф свадебная фотосъемка	
фотосессия воронеж цены	фотограф свадьбу день	детская фотосессия воронеж
фотограф цены		детские фотографы воронежа
		детский фотограф воронеж
подарочный сертификат на фотосессию		фотосессия детей воронеж
сертификат на фотосессию		детские семейный фотограф
фотографы воронежа отзывы		фотограф новорожденного
		фотограф новорожденных
		фотосессия новорожденных
		фотосессия новорожденного
		выписка из роддома фотосессия
		фотограф на выписку
		фотосессия выписка

Интересно

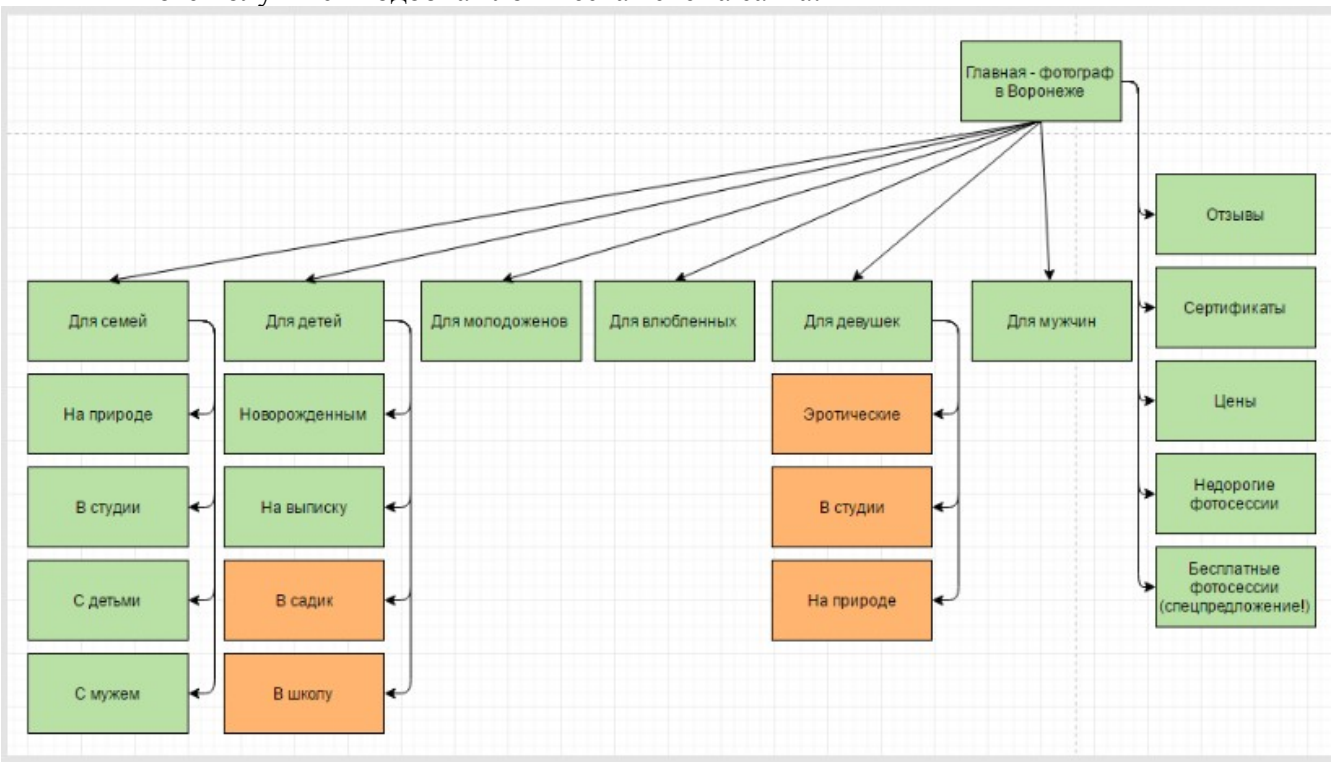
Кстати, часто бывает так, что логика еще на этапе кластеризации подводит сеошников. Появляются такие интересные идеи, как «ну тут же тоже есть про дверь, вот я и думаю – лишним не будет», это когда речь идет про ключи вроде «купить металлические двери» и «двери купить». Запомните, если в ключевой фразе есть какой-то конкретный параметр (металлический, на свадьбу, с рюшами), то такие слова нужно отделять от слов, которые никаких параметров не имеют. Ключи без параметров всегда будут уровнем выше ключей с параметрами.

Итак, у нас получилось несколько групп:

- Недорогие фотосессии
- Бесплатные фотосессии
- Общие запросы типа «Фотограф Воронеж»
- Стоимость услуг фотографов
- Подарочные сертификаты
- Отзывы
- Фотосессия в студии
- Для мужчин
- На свадьбу
- Семейный фотограф
- Семейная фотосессия на природе
- Семейная фотосессия в студии
- Фотосессия с детьми
- Фотосессия с мужем
- Фотосессия для беременных
- Детский фотограф
- Фотосессия для новорожденных
- Фотосессия на выписку.

Итак, СЯ кластеризовали. Что делать дальше? Начинаем создавать структуру продающего сайта. Учитываем, что какие-то группы могут подчиняться одна другой. Например, фотосессия с мужем относится к семейным фотографиям, соответственно эту группу лучше подчинить группе «семейный фотограф». Логика здесь может быть разная, сделайте так, как считаете правильным. Например, вам кажется, что лучше выделить группу «детский фотограф» в верхний уровень и в подчинение ему отдать группы «новорожденным», «на выписку», «садик» и т.д. Хотя так же логично было бы в верхний уровень выделить страницу «фотограф на торжество», и страницу «выписка» отдать в подчинение уже этой странице, наряду со страницами «свадьба», «день рождения» и подобное.

В итоге получится подобная логическая схема сайта:



Здесь отдельным столбцом справа расположились страницы с информацией (что и как). Такие страницы чаще всего помещаются в верхнем горизонтальном меню, а

остальные группы уже показывают товар лицом и рассказывают о том, на что способен фотограф. Оранжевым выделены те страницы, которые не предусматриваются нашим семантическим ядром, но при этом могут быть по желанию заказчика. Их так же нужно учитывать при создании структуры сайта. Теперь у нас есть для каждой группы запросов отдельная страница, которую можно успешно продвигать.

Важно!

Если у вас в работе группы запросов одного порядка (тема праздников - свадьба, выписка, выпускной), вы можете не придумывать особой структуры с подуровнями – просто сделайте все в подчинение главной странице. Если же вы видите, что у вас группы запросов разного плана (свадьба, в лесу, с мужем), продумайте, как можно упорядочить такие группы, чтобы это не стало кашей на сайте. Выделите группы «с кем», «где», «когда» и в каждую группу уже добавляйте следующим уровнем страницы с конкретикой. С кем? С мужем. С детьми. С лошадью. С подругами.

Даже если для страницы «с кем» у вас нет группы ключевых слов. Это и будет грамотная структура сайта.

Как посмотреть структуру сайта конкурента?

В продвижении сайтов очень часто обращаются к анализу конкурентов и в общем-то структуру сайта анализировать тоже полезно. Только поняв, как конкурент достиг успеха, вы сможете сделать свой сайт успешным, используя его находки и избегая ошибок.

Чтобы понять, какая же структура у чужого сайта, вы можете проверить ее вручную.

Зайдите на сайт и оцените сначала визуально, какие страницы и разделы могут присутствовать у конкурента. Чтобы понять, действительно ли визуальные элементы имеют какую-то иерархию в структуре сайта, обратите внимание на URL страниц. То есть, если вы видите, что у страницы «Проемы» в меню есть подстраница «Расширение проемов», это еще не значит, что вторая страница подчиняется первой в структуре сайта (визуально элементы сайта могут располагаться как угодно, это не показатель как таковой, просто чаще все же структура отображена и в визуальной составляющей сайта). Чтобы узнать, так ли это на самом деле, посмотрите на урл страницы «Расширение проемов», если в адресе страницы вы увидите папку «Проемы», значит, она действительно по структуре подчиняется этой странице: site.ru/proemy/rasshirenie-proemov. Если вы видите урл вида site.ru/rasshirenie-proemov – то в структуре эта страница сама по себе, что неправильно.

Кроме самостоятельного изучения структуры, возможно использование сервисов и программ, но к ним стоит относиться осторожно, не все они работают корректно.

Содержание отчета

1. Цель
2. Для чего нужна структура сайта
3. Требования к созданию структуры сайта
4. Семантическая схема сайта
5. Выводы

Контрольные вопросы

1. Для чего нужна структура сайта
2. Анализ конкурентов при создании структуры сайта
3. Семантическая схема сайта
4. Виды структуры сайта
5. Как создать структуру своего сайта. Виды сайтов и их структура
6. Проектирование структуры сайта.
7. Где и как нарисовать структуру сайта?
8. Как посмотреть структуру сайта конкурента?

Лабораторная работа №№ 20

Разработка эскизов веб-приложения

Цель: *получить практические навыки разработки эскиза веб-приложения.*

Ход работы:

1. Изучить теоретический материал.
2. Выполнить задания в соответствии с указаниями.
3. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
4. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Прототип – это первоначальный набросок сайта на бумаге. Во время разработки сайта для компьютеров или мобильников такие рисунки помогают в выборе точек фокуса внимания и расстановке элементов дизайна перед началом работы. **Эскизы мобильных - и веб-шаблонов** помогают сконцентрироваться на креативной составляющей проекта. Кроме того, это лучший для клиента способ понять, за что же он собирается платить. Наброски шаблона помогают дизайнеру не сбиться с пути во время работы над сайтом. Также они экономят деньги и время, которые потребовались бы для корректировки первоначальных набросков онлайн.

Прототипы могут быть выполнены с различной степенью детализации. Карандашные эскизы являются первым шагом в концептуализации проекта. В данном случае менее детальный рисунок будет более предпочтительным для разработчиков. Фактически такие эскизы, выполняемые для мобильных и веб сайтов, наиболее популярны. Низкокачественные каркасные эскизы позволяют отсеять ненужные элементы. С помощью них дизайнер может подготовиться к работе над проектом еще до того, как приступит к написанию кода. Также они позволяют сравнить конкретный проект с аналогами.

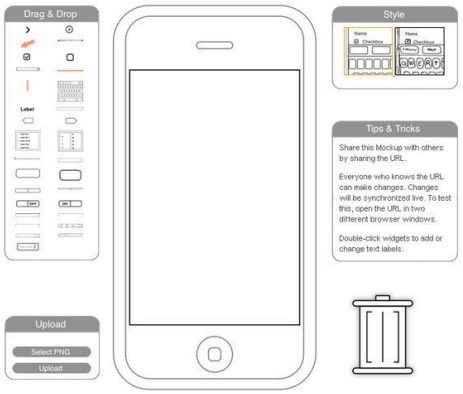
Веб- и мобильные наброски важны с точки зрения поиска инвесторов для проекта. Имея в своем распоряжении начальный эскиз шаблона, вам потребуются считанные секунды, чтобы определить круг вероятных инвесторов и заинтересованных лиц. Карандашные эскизы являются первым шагом в продвижении идей и налаживании связей с привлекающими вас инвесторами. Есть несколько примеров сделок, заключенных сразу же после того, как инвесторы ознакомились с эскизами сайта и посчитали проект достаточно прибыльным и привлекательным. Для того, чтобы всегда быть первым в конкурентной борьбе, очень важно уметь рисовать эффектные карандашные прототипы веб- и мобильных сайтов.

Если ваша цель – разработка качественного сайта и точная его настройка, вам непременно стоит обратить внимание на прототипы для достижения наилучших результатов

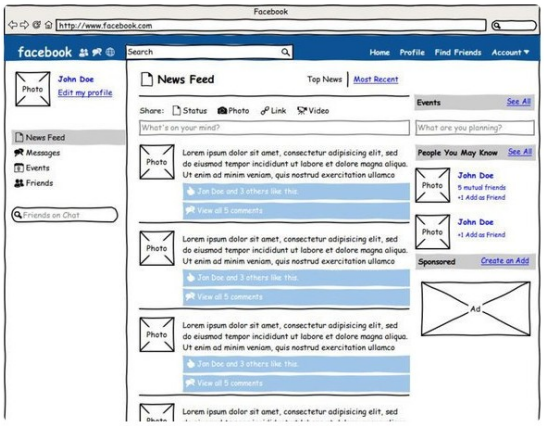
Программы для создания эскизов позволяют быстро создавать наброски веб-страниц, планировать переходы, создавая связи между элементами страницы и эскизом страницы, предоставляют возможность совместной работы над эскизами. Программы представляют собой веб-приложения.

Ресурсы макетов и прототипов являются очень полезными для веб-дизайнеров, когда нужно создать эскиз, чтобы показать клиенту, как их дизайн будет выглядеть, когда будет полностью разработан. С помощью таких инструментов веб-дизайнеры могут быстро создать макет своего веб или мобильного проекта и затем более эффективно общаться со своими клиентами.

Iphone market. Это бесплатный инструмент специально разработанный для создания мобильных макетов. Все внесенные изменения отображаются в реальном времени.



WireframeSketcher. Этот инструмент позволяет создавать макеты, эскизы и прототипы для мобильных, настольных и веб-приложений.



Proto.io. С помощью этого платного инструмента, можно легко создавать прототипы интерфейса для мобильных и планшетных приложений. Вы можете создавать проекты для iPad или iPhone.

iMockups. iMockups – это платный мобильный инструмент, который позволят создавать макеты приложений для iPhone, IPad и веб-проектов.

Adobe Proto. Adobe Proto это еще один платный мобильный инструмент, который позволяет создавать легкие и интерактивные эскизы для мобильных и веб приложений.

А Вы использовали какие-то из указанных инструментов раньше? Оставьте нам комментарий и дайте знать ваше мнение об этой коллекции. Также, если вы знаете другие полезные ресурсы, то мы будем признательны, если вы поделитесь ними с нашими читателями.

А также:

- http://iplotz.com/ - есть desktop-версия
- http://www.balsamiq.com/
- http://www.justproto.com/en/
- http://www.protoshare.com/
- https://www.jumpchart.com/
- https://pidoco.com/
- http://mocksup.com/
- http://mockflow.com/ - есть desktop-версия
- https://cacoo.com/
- http://www.mockuptiger.com/

Практическая часть

Задание. Разработайте эскизы веб-приложения в соответствии с индивидуальным вариантом, основываясь на нижеприведенном примере. Используйте любую доступную программу для создания эскизов веб-приложения.

Содержание отчета

1. Цель
2. Этапы создания эскизов веб страниц
3. Результаты работы
4. Выводы

Контрольные вопросы

1. Что такое эскиз веб-страницы?
2. Назначение эскизов веб-страниц

3. Программы для создания эскизов

Лабораторная работа № 21

Разработка прототипа дизайна веб-приложения

Цель: *получить практические навыки разработки прототипа дизайна веб-приложения.*

Ход работы:

1. Изучить теоретический материал.
2. Выполнить задания в соответствии с указаниями.
3. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
4. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Прототип – это визуальное представление будущего ресурса. Указываются значимые элементы дизайна, уделяется внимание вёрстке, детально прорисовывается расположение кнопок, модулей и главных блоков. Чем точнее будет составлен примерный портрет, тем лучше получится ресурс. Лучше всего использовать бесплатную программу для создания прототипа сайта.

Готовый эскиз позволит придерживаться определённой структуры. К тому же, он всегда будет под рукой.

Приведу несколько причин, почему это необходимо:

- Наглядное представление – каркасное моделирование расставит все страницы по местам с указанием отдельных блоков на каждой из них;
- Планирование – есть возможность наметить шаги реализации и дальнейшего развития ресурса. Используя программу для создания прототипа сайта бесплатно и на выгодных условиях, вы сможете поэкспериментировать над отображением отдельных блоков;
- Экономия времени – предварительный рисунок позволит рационально распределить собственные ресурсы;
- Определение работы дизайнера – детальное отображение элементов помогает при дальнейшей разработке дизайна.

Условимся, что прототип выступает в роли предварительного эскиза или наброска, демонстрирующего основные достоинства и отличия портала. Такая зарисовка снижает риск возникновения ошибок. Для достижения качественного результата воспользуйтесь программами для создания прототипа сайта на русском языке.

Виды прототипов

Существует несколько разновидностей, каждый из которых имеет свои особенности:

- Бумажный – простой и незатейливый способ быстро отобразить все необходимые элементы предполагаемого проекта;
- Программный – обладает эстетичным видом и возможностью гибких правок;
- Динамический – производится посредством кодировки, так предоставляется точное отображение готовящегося к запуску ресурса.

Все эти виды по-своему удобны. Для того, чтобы эскиз был наилучшим и отображал главные особенности портала в деталях, используйте программу для создания прототипа сайта. Такой софт направлен на воплощение удобной схемы расположения блоков с целью дальнейшего корректирования.

Программы для создания прототипа

1. Cacoo – красочный сервис с макетами для использования. Лёгкая версия в режиме онлайн. Программа для создания прототипа сайта на русском имеет красочные элементы. Полученный проект будет обладать оригинальностью и креативностью. Результат экспортируется в формате в png;

2. «Moqups» – прекрасная альтернатива предыдущему аналогу. Встроенные диаграммы и схемы позволяют создать эскиз на многостраничный портал. Предоставляется объединение страниц со ссылками, добавление изображений, а также экспорт файлов;

3. HotGloo – полезное приложение для отрисовки макета в режиме онлайн. Достаточно просто перетаскивать элементы, вставлять тематические картинки, изменять масштаб деталей, объединять в единую концепцию. Также есть полезная функция переименования файлов;

4. Gliffy – простой интерфейс и продуманная блок-схема являются главными преимуществами. Программа для создания прототипа сайта содержит множество инструментов для реализации каркаса страниц, после чего они связываются между собой;

5. Axure – функциональное приложение для представления диаграмм и схем прототипирования. Здесь можно разработать карты сайтов, проекты на тему бизнеса, организовать подробную структуру портала, построить примерный каркас в виде связанных разделов;

6. Mockup Builder – простой софт с удобной навигацией позволяет за несколько часов сделать небольшие наброски, которые будут выглядеть презентабельно и стильно. Созданный проект можно легко экспортировать, поддерживаемый формат pdf и png;

7. Creately – мощный редактор с полезными опциями. Работает напрямую через браузер. Отзывчив к диаграммам и сложным схематическим конструкциям;

8. MockFlow – программа для создания прототипа сайта заточена под редактирование элементов в режиме реального времени. Обширная библиотека шаблонов и графики. Помогает разработать проекты различной сложности с дальнейшим экспортом.

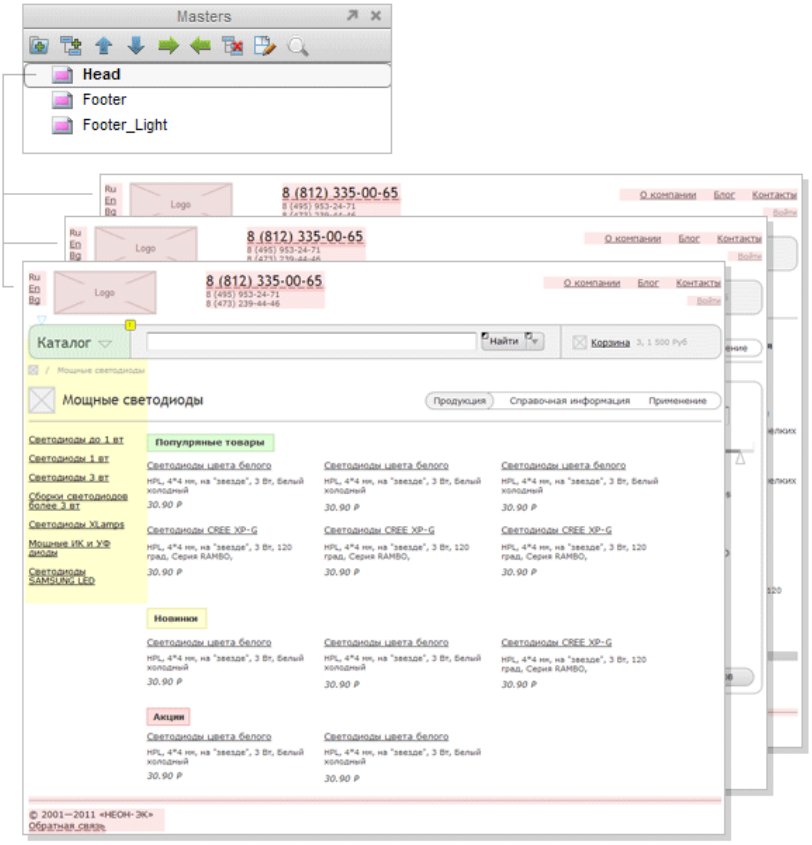
Подобные сервисы обладают богатым функционалом, с помощью которого рисуются макеты любой сложности.

Благодаря им без труда можно отобразить эскиз будущего многостраничного портала или небольшого корпоративного.

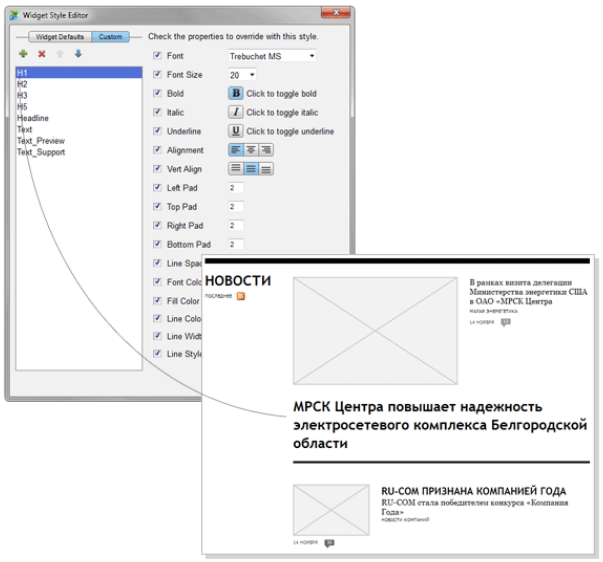
Разработка прототипа поможет определиться с разделами и блоками на страницах портала. Наглядный рисунок структуры выступает в роли вспомогательного инструмента при построении ресурса. Используйте проверенные софты для реализации своих идей.

Глубина проработки

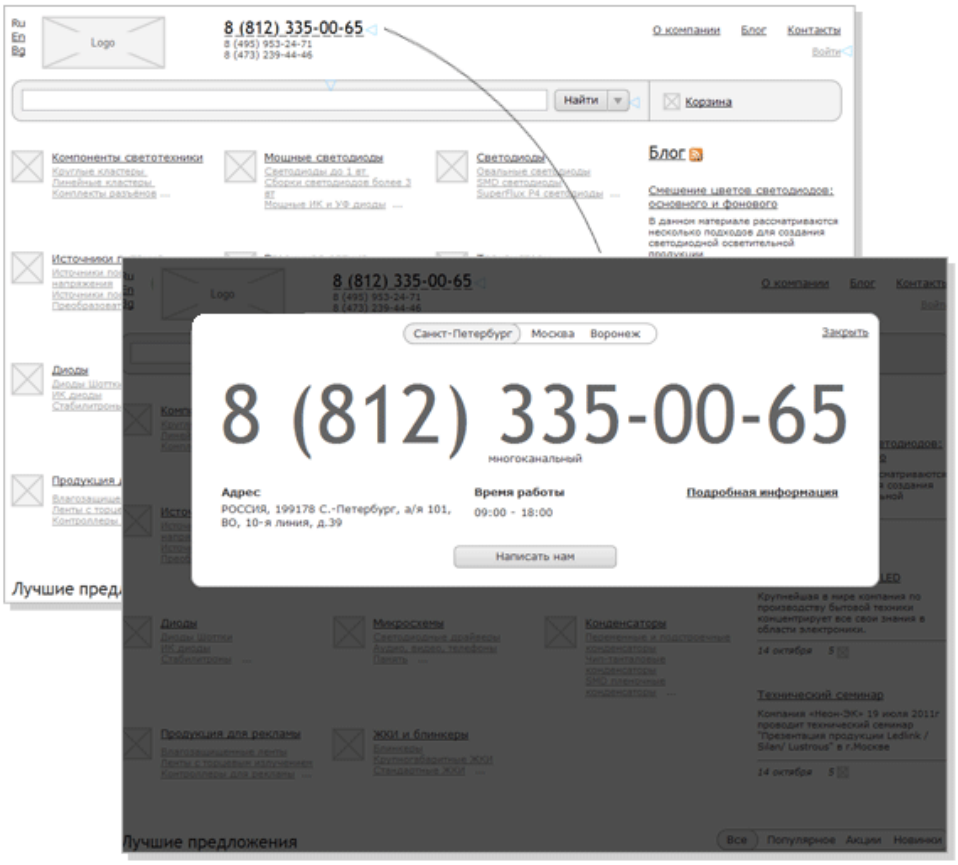
1. Все повторяющиеся элементы должны быть сделаны через мастера. Мастера должны быть разложены по папкам и поименованы в соответствии со своим предназначением.



Обязательно использовать встроенный редактор стилей — все стили текста в прототипе задаются только через него. Чем стилей меньше, тем лучше, отталкиваемся от семантики текстов и стандартных HTML-тегов: h1, h2 и т.д.

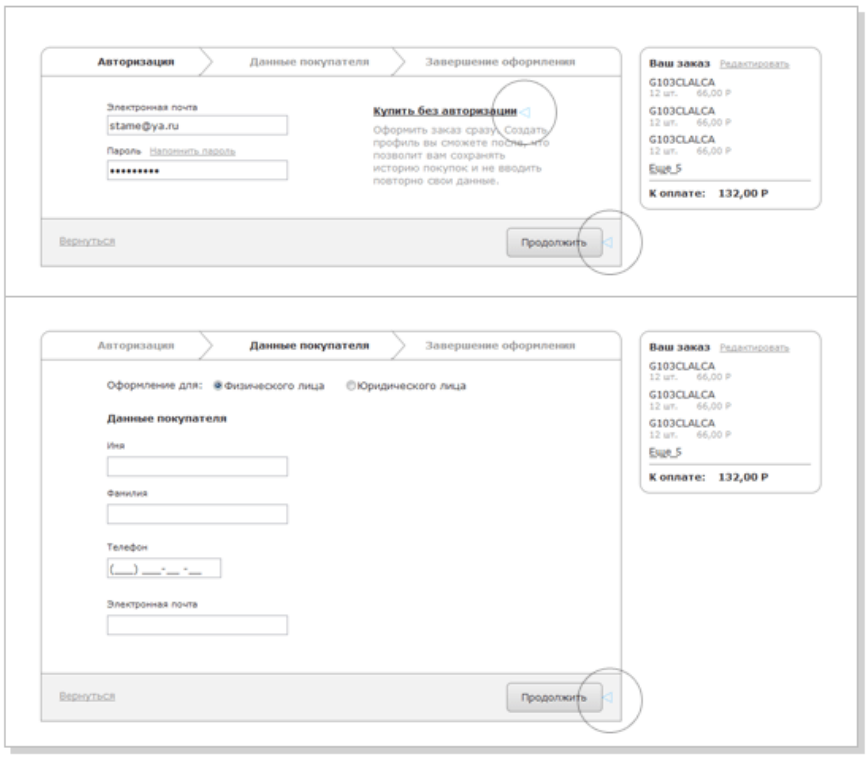


3. Интерактивные элементы нужно программировать и, если срабатывание не очевидно, то помечать их для заказчика специальным знаком — у нас принят



треугольник.

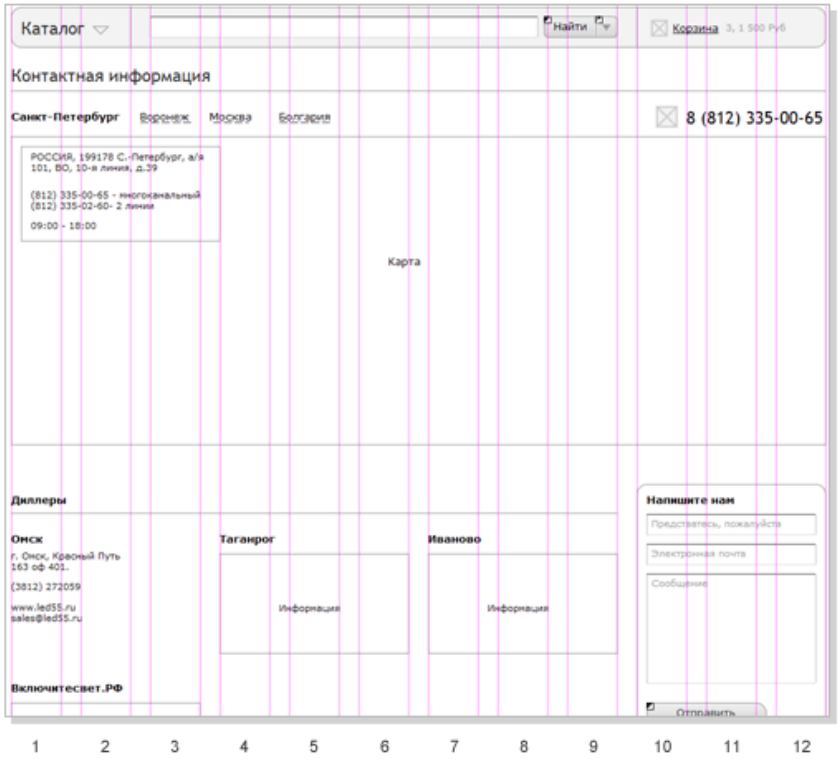
4. Необходимость залинковки прототипа обсуждаем с менеджером проекта. Если прототип залинкован частично, то отмечаем элементы, на которые можно нажать, треугольником. Не забываем подчеркивать псевдоссылки пунктирной линией.



5. Однотипные разделы можно не дублировать, показывая один раз. Тем не менее, если для них есть контент, размещение которого необходимо планировать, то делаем все страницы.

Внешний вид

6. В основе прототипа должны быть сетки. Удобнее всего использовать классические сетки 12/16 колонок, но лучше всего согласовывать это с арт-директором, который будет курировать дизайн. Все объекты прототипа располагаются по сетке, далее дизайнеры сохраняют такое размещение.

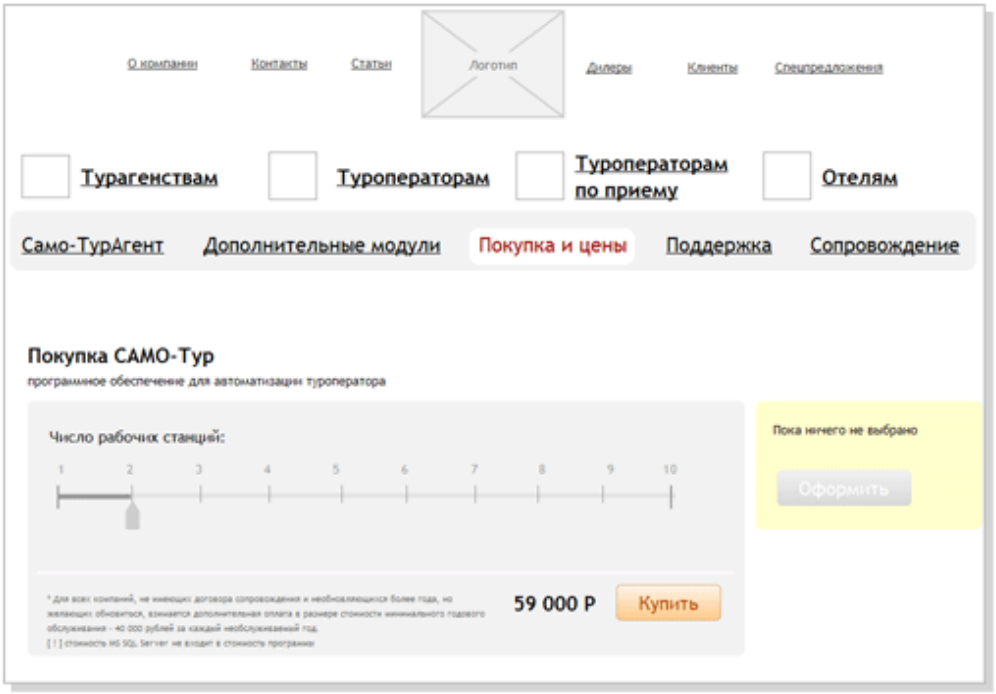


7. При наличии реальных текстов, их можно и нужно использовать в прототипе. Запрещено использовать Lorem Ipsum и другие тексты, не имеющие отношения к проекту.

8. Небольшие фрагменты текста: подписи к кнопкам, заголовки и т.д. пишем сами. Эти задачи не требуют участия копирайтера, менеджера и тем более клиента.

9. Все должно быть просто и аккуратно: минимум цветов, стандартные веб-шрифты, отсутствие полноцветных изображений и т.д.

10. Используйте осмысленное цветовое кодирование в прототипе.



11. Прототип — это инструмент, который позволяет проработать и утвердить функциональность проекта отдельно от дизайна. Поэтому он ни в коем случае не должен

выглядеть как дизайн. Запрещено тратить время на всевозможные красивости.

12. Используем стандартные контролы, если нет острой необходимости в обратном.

Другое

12. Файлы, страницы, мастера и пр. необходимо именовать только латиницей. При выгрузке прототипа на сервер, не работает все, где использована кириллица.

13. Все имена в прототипе должны быть осмыслены, начинаться с большой буквы и отражать суть объекта, пробелы заменяются на подчеркивание. Например: Cart_active_var2

Практическая часть

Задание. Разработайте бумажные и программные прототипы дизайна веб-приложения в соответствии с индивидуальным вариантом. Используйте любую доступную программу для создания прототипов дизайна веб-приложения.

Содержание отчета

1. Цель
2. Этапы создания прототипов веб страниц
3. Результаты работы
4. Выводы

Контрольные вопросы

1. Что такое прототип веб-страницы?
2. Назначение прототипов веб-страниц
3. Программы для создания прототипов

Лабораторная работа № 22

Разработка схемы интерфейса веб-приложения

Цель: *получить практические навыки разработки схемы интерфейса веб-приложения.*

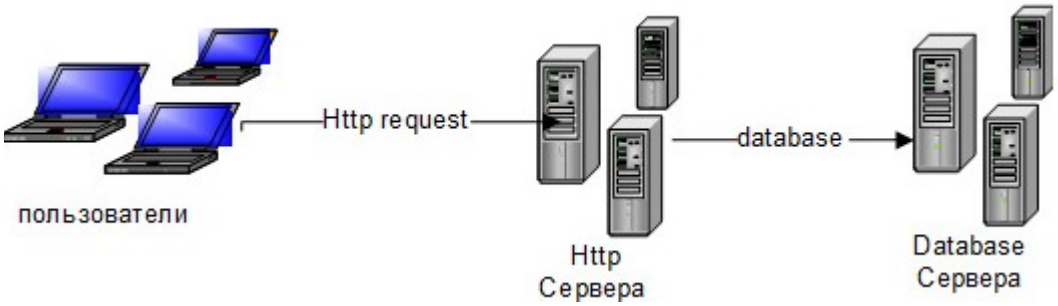
Ход работы:

1. Изучить теоретический материал.
2. Выполнить задания в соответствии с указаниями.
3. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
4. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Проектирование корпоративного веб приложения, как и любого другого приложения, стоит начать с определения первоначальный целей и области решаемых задач. Создать реестр заинтересованных лиц. На следующем этапе необходимо собрать требования к приложению, которое необходимо разработать. Уточнить цели и область решаемых задач и построить иерархическую структуру работ

Рассмотрим отдельно задачу построения иерархической структуры работ. Каждое web-приложение можно представить в следующем виде:



Другими словами, каждое web-приложение отправляет http запросы на web-сервер для получения полезных данных. Программа под управлением web-сервера использует ту или иную модель для хранения данных. В современном мире чаще всего используются базы данных, SQL или NoSQL.

Формально каждое web-приложение можно разбить на 3 взаимно независимые части.

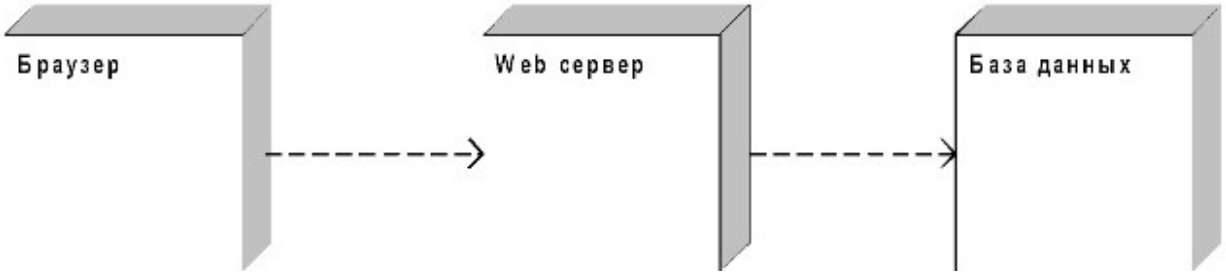
Модуль, который исполняется WEB-браузером. Это приложение может быть написано на любом языке, который поддерживает браузер. Чаще всего используется язык JavaScript, как наиболее поддерживаемый и имеющий большую библиотечную поддержку. Это очень важно, так как позволяет существенно экономить бюджеты проектов.

1. Модуль, исполняемый на серверной стороне под управлением web-сервера. Это приложение может быть написано на любом языке, интерпретацию которого поддерживает выбранный Вами web-сервер. Последнее время, часто, в качестве языка программирования выбирается язык Java. Этот язык также имеет серьезную библиотечную поддержку.

2. База данных. В этой области так же существует достаточно широкий выбор. Есть промышленные базы данных, такие как Oracle, DB2, PostgreSQL. Есть легкие базы данных, такие как MySQL. База данных выбирается основываясь на целях и области решаемых задач.

Возможные эталонные модели проектирования web-приложений.

При построении архитектуры web — приложения необходимо максимально уменьшить зависимость между структурными единицами. В общем случае приложение состоит из трех структурных единиц.



1. Модуль, который работает под управлением браузера.
2. Модуль, который работает под управлением web-сервера.
3. База данных.

Эти структурные единицы порождают два вида связей.

1. Связь между браузером и серверной частью.
2. Связь между серверной частью и базой данных.

Для достижения цели максимальной независимости между структурными единицами, необходимо чтобы каждая структурная единица оперировала только необходимым ей набором данных. Рассмотрим более подробно.

Браузер — это прикладное программное обеспечение для просмотра web страниц.

HTML – это стандартный язык разметки документов. Большинство современных web-браузеров способны интерпретировать язык HTML.

Web сервер — это программное обеспечение, которое способно принимать HTTP запросы от клиентов, обрабатывать их и отправлять ответ в соответствии со стандартом протокола.

База данных — это представленная в объективной форме совокупность самостоятельных материалов, систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью ЭВМ.(Wiki)

Минимизация зависимостей

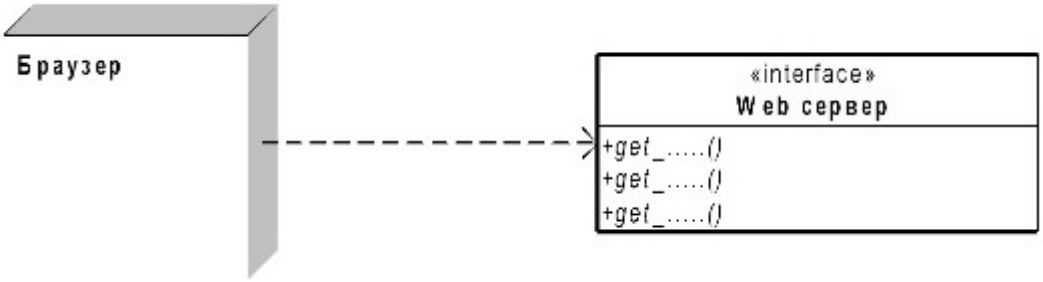
Для минимизации зависимостей между «Браузером» и Web-сервером необходимо, чтобы язык разметки HTML был задействован только в браузере, а Web-сервер предоставлял интерфейс для получения необходимых данных для страницы.

Для решения этой задачи необходимо:

Определить цели и область решаемых задач, которые будут решаться в рамках создаваемого интерфейса.

- Определить API серверной части.
- Выбрать протокол взаимодействия между серверной и клиентской частью. Создание протокола удобнее всего выбрать на базе XML, так как большинство современных браузеров имеют встроенную поддержку этого языка.
- Написать документ, в котором будет изложен протокол.

Наша диаграмма может быть преобразована в следующий вид:



Далее «Браузер» преобразуется в UML диаграммы состояний. На этих диаграммах будет отражено, в каком случае вызывается тот или иной метод.

Данная модель достижима двумя путями

1. Программа выполняемая «Браузером» написана на JavaScript и общается с Web-Сервером через AJAX, получая ответы в соответствие с определенным протоколом.
2. «Браузер» интерпретирует только HTML код, а преобразования происходят посредством XSLT преобразований на стороне Web-Сервера.

В каждом из этих случаев достигается разделение программной части Web-Сервера и «Браузера». Т.е используя данную модель возможно вносить изменения в структурную единицу для «Браузера» и не вызывать косвенных изменений в серверной части. Это очень важно, так как ведет к уменьшению затрат на обработку запросов на изменения. Это происходит в силу того, что изменения в одной структурной единицы не выходят за ее рамки.

Взаимодействие Web-Сервера и Базы данных

Взаимодействие базы данных и web-сервера возможно организовать на основании двух принципиально разных сценариях:

1. Бизнес логика находится в базе данных.
2. Бизнес логика находится в коде web-сервера.

В первом случае база данных хранит данные и предоставляет интерфейс доступа к данным:

1. Выборка данных — решается через представления.
2. Модификация данных — решается через хранимые процедуры.

Программа для web-сервера является драйвером для доступа к бизнес-логике. Т.е она просто связывает Браузер с бизнес логикой, которая реализована в базе данных.

Во втором случае база данных хранит данные, и предоставляет прямой доступ к данным. Бизнес-логика реализована в коде web-сервера. В этом случае база данных предоставляет транзакции для проведения атомарных операций.

Для минимизации зависимостей между Web-Сервером и Базой данных, необходимо, чтобы бизнес-логика была определена только в одном месте. Т.е либо в коде Web-Сервера, либо в Базе данных. Это очень важно, так как ведет к уменьшению затрат на обработку запросов на изменения. Это происходит в силу того, что изменения в одной структурной единицы не выходят за ее рамки.

Иерархическая структура работ

На основании изложенного выше материала иерархическая структура работ примет следующий вид:

1. Модуль для «Браузера».
2. Модуль для Web-Сервера.
3. Модуль для Базы данных.
4. Протокол обмена между модулем «Браузера» и Web-Сервером.
5. Интерфейс взаимодействия между модулем «Браузера» и Web-Сервером.
6. Интерфейс взаимодействия между Web-Сервером и Базой данных.

Практическая часть

Задание. Разработайте схему-интерфейса веб-приложения в соответствии с индивидуальным вариантом. Используйте любую доступную программу для создания схемы-интерфейса веб-приложения.

Содержание отчета

1. Цель
2. Этапы создания схемы интерфейса веб-приложения
3. Результаты работы
4. Выводы

Контрольные вопросы

1. Что такое схема интерфейса веб-приложения?
2. Назначение схем интерфейса веб-приложения
3. Программы для создания схемы интерфейса веб-приложения.

Лабораторная работа № 23

Пользовательский интерфейс средствами CSS

Цель: *получить практические навыки разработки пользовательского интерфейса веб-приложения средствами CSS.*

Ход работы:

1. Изучить теоретический материал.
2. Выполнить задания в соответствии с указаниями.
3. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
4. Подготовить ответы на контрольные вопросы (устно)

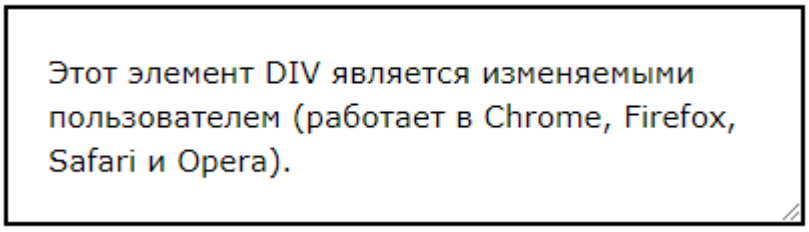
Теоретический материал:

CSS Пользовательский интерфейс

CSS3 имеет новые возможности пользовательского интерфейса, такие как изменение размеров элементов, контуры, и проклейки коробки.

CSS3 Resizing

resize свойство определяет , должен ли элемент быть изменяемыми пользователем.



Следующий пример позволяет пользователю изменять размер только ширину элемент:

```
div {
    resize: horizontal;
    overflow: auto;
}
```

Следующий пример позволяет пользователю изменять размер только высоту элемент:

```
div {
    resize: vertical;
    overflow: auto;
}
```

Следующий пример позволяет пользователю изменять размер как высоту и ширину элемент:

```
div {
    resize: both;
    overflow: auto;
}
```

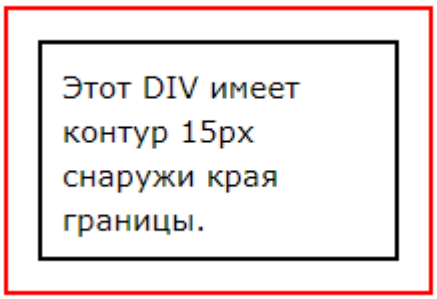
CSS3 Outline Смещение

outline-offset свойство добавляет пространство между контуром и краем или границы элемента.

Контуры отличается от границ тремя способами:

- Контур линия, нарисованная вокруг элементов, за пределами края границы

- Контур не занимает пространство
- Контур может быть непрямоугольная



В следующем примере используется свойство контура смещения, чтобы добавить 15px пространство между границей и набросков:

пример

```
div {
    border: 1px solid black;
    outline: 1px solid red;
    outline-offset: 15px;
}
```

Свойства пользовательского интерфейса CSS3

В следующей таблице перечислены все свойства пользовательского интерфейса

Свойство	Описание
box-sizing	Позволяет включать отступы и границы в общей ширины и высоты в Элементом
nav-down	Указывает, где для навигации при использовании навигационной клавиши со стрелкой вниз
nav-index	Определяет порядок табуляции для элемента
nav-left	Указывает, где для навигации при использовании стрелки левой навигационной клавиши
nav-right	Указывает, где для навигации при использовании стрелки правой клавиши навигации
nav-up	Указывает, где для навигации при использовании со стрелкой вверх навигационной клавиши
outline-offset	Добавляет пространство между контуром и краем или границы элемента
resize	Определяет, является ли изменяемыми пользователем элемент

CSS свойства пользовательского интерфейса.

Свойство **cursor** определяет вид указателя мыши, расположенного над текущим элементом.

Значения:

auto - определяется браузером.
crosshair - крестик.
pointer - указующий перст.
text - текстовый указатель.

wait - песочные часы.
help - стандартный указатель со знаком вопроса.
URL - ссылка на нестандартный указатель.
e-, ne-, nw-, n-, se-, sw-, s-, w-resize - стрелки для перемещения.

Пример:

```
.p { cursor : pointer; }
```

Существует возможность управлять линейками прокрутки.
Данный перечень свойств позволяет это делать.

Свойства:

scrollbar-3dlight-color - Определяет или устанавливает цвет верха и левой части ползунка и кнопок со стрелками на полосе прокрутки.
scrollbar-arrow-color - Устанавливает или определяет цвет стрелок на кнопке со стрелками.
scrollbar-base-color - Устанавливает или определяет цвет основных элементов ползунка: ползунка, кнопок со стрелками, дорожки для ползунка, если не определены параметры в scrollbar-face-color.
scrollbar-darkshadow-color - Устанавливает или определяет цвет тени для ползунка и кнопок со стрелками.
scrollbar-face-color - Устанавливает или определяет цвет ползунка и кнопок со стрелками. Также, если вы не задали параметр SCROLLBAR-TRACK-COLOR, у вас изменится цвет дорожки.
scrollbar-highlight-color - Устанавливает или получает цвет подсветки, создающий эффект объёмности.
scrollbar-shadow-color - Схоже с scrollbar-darkshadow-color.
scrollbar-track-color - Устанавливает или получает цвет дорожки для ползунка.

Пример:

```
body { scrollbar-face-color : red; }
```

Практическая часть

Задание. Разработайте пользовательский интерфейс страниц веб-приложения средствами CSS в соответствии с индивидуальным вариантом.

Содержание отчета

1. Цель
2. Результаты работы
3. Выводы

Контрольные вопросы

1. Правила разработки пользовательского интерфейса?
2. Свойства CSS, определяющие пользовательский интерфейс