Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Томский государственный университет систем управления и радиоэлектроники»

Кафедра экономической математики, информатики и статистики

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В ДИЗАЙНЕ

Методические указания к практическим и самостоятельным работам

Томск 2018

Безрук Анна Викторовна

Информационные технологий в дизайне методические указания по практическим и самостоятельным занятиям для студентов дневной заочной формы обучения.

Целью данного пособия является пошаговое изучение HTML программирования. Разработка охватывает основные темы при создании HTML страниц.

Содержание

ПРАКТИЧЕСКАЯ РАБОТА №1
СРАВНИТЕЛЬНЫЙ АНАЛИЗ САЙТОВ ПО ЗАДАННЫМ КРИТЕРИЯМ

1 Цель рабы

Целью данной работы является анализ сайтов учебных заведений по заданным критериям с выбором наилучшего и наихудшего из кандидатов.

Для рассмотрения выбрано три сайта предложенных преподавателем:

пример

1. Томская банковская школа (http://www.tbs.tomsk.ru);
2. Томский финансово-юридический техникум (http://tomfut.tomsk.ru);
3. Томский коммунально-строительный техникум (http://tomkst.tomsk.ru);
4. Томский индустриальный техникум (http://www.tomintech.ru/lyceum/index.php?rm=news&action=lentList&cat_id=283)

Таблица 1 - Сравнение по основным критериям

Название техникума			
Критерий	Томский финансово юридический техникум	Томский коммунально строительный техникум	Томский индустриальный техникум
время загрузки сайта (сек)	0,26	0,19	3,5
размер первой страницы (кб)	73	146	92
уровень домена	3	3	3
дата создания и последнего обновления	2013-2017	2006-2017	2004-2017

технология создания сайта (https://builtwith.com)	jQuery, PHP	jQuery, PHP	PHP, Silverlight, MooTools

Продолжение таблицы 1

видимость поисковых систем	Google (первый в списке) Yandex (первый в списке) Mail (второй в списке)	Google (первый в списке) Yandex (первый в списке) Mail (первый в списке)	Google (первый в списке) Yandex (первый в списке) Mail (тринадцатый в списке)
дизайн в различных браузерах	В Google Chrome 61, Firefox 50, IE11 и Opera 40 все одинаково	В Google Chrome 61, Firefox 50, IE11 и Opera 40 все одинаково	В Google Chrome 61, Firefox 50, IE11 и Opera 40 все одинаково
единый стиль	нет	нет	нет
семейство шрифтов	PT Sans (не безопасный)	Georgia (безопасный)	Georgia (безопасный)
размер шрифта (px)	16	13	13
цветовая палитра	4 цвета	4 цвета	3 цвета
звук	нет	нет	нет
горизонтальная прокрутка	нет	нет	нет
точки в заголовках	нет	нет	нет
карта сайта	нет	нет	да
движущиеся надписи	нет	нет	нет
доступ к информации в три клика	да	да	да
"хвостатые" ссылки	нет	нет	нет

ссылка на главную страницу в левом верхнем углу	да	нет	да

Окончание таблицы 1

подписи картинок	да	да	нет
баннерная реклама	нет	нет	да
декоративные элементы, отвлекающие глаз	нет	нет	да
оформление подвала	контакты, социальные сети, создатели	контакты, дополнительная информация о заведении, создатели	контакты, данные Яндекс метрики, html и css код
наличие мобильной версии	нет, но сайт изменяет свои пропорции, адаптируясь под мобильный экран	нет, но сайт изменяет свои пропорции, адаптируясь под мобильный экран	нет
наличие интеграции с социальными сетями	да	нет	нет
уникальность контента	да	да	да
.Seo оптимизация (app.rankval.com)	76%	79.2%	75%

Ссылки вывод по каждому сайту.

Заключение

выбор наилучшего и наихудшего из сайтов. Вывод о проделанной работе.

ПРАКТИЧЕСКАЯ РАБОТА №2
ЯЗЫК HTML. СТРУКТУРА ДОКУМЕНТА.

Цель работы: знакомство с языком гипертекстовой разметки HTML; средства разработки документов HTML; структура гипертекстового документа, основные части; основные метки и атрибуты тела HTML-документа.

Теоретическая часть:

Чтобы опубликовать документ (здесь и далее под документом понимается файл, содержащий некоторую информацию) на Интернет, достаточно поместить его на сервер, постоянно подключенный к Интернет и способный общаться с другими серверами с помощью протокола передачи гипертекстов (HyperText Transfer Protocol, или http://). Совокупность таких серверов получила название "всемирной паутины" (World Wide Web, или WWW).

Что такое HTML?

Термин HTML (HyperText Markup Language) означает "язык маркировки гипертекстов". Первую версию HTML разработал сотрудник Европейской лаборатории физики элементарных частиц Тим Бернерс-Ли.

Со времени создания первой версии HTML претерпел некоторые изменения. Как и многое другое в компьютерном мире, версии, или спецификации, HTML оказались пронумерованными. Известны спецификации 2.0, 3.0, 3.2 и 4.0. Текущую спецификацию HTML всегда можно найти на сервере W3C (http://www.w3.org/).

Для освоения HTML Вам понадобятся две вещи:

1.Любой браузер, т.е., программа, пригодная для просмотра HTML-файлов.

2.Любой редактор текстовых файлов, поддерживающий русский язык в выбранной Вами кодировке. Если на Вашем компьютере установлен Windows, вполне подойдет Notepad.

Мы будем использовать текстовый редактор Notepad для подготовки HTML-файлов, а браузер MicroSoft Internet Explorer (MSIE) — как инструмент контроля за сделанным.

Свои первые HTML-файлы Вы будете разрабатывать у себя на локальном диске. Другими словами, компьютер, на котором Вы будете заниматься, может и не иметь подключения к Интернет. При этом один и тот же *.html-файл может быть одновременно открыт и в текстовом редакторе, и в браузере.

Сохранив изменения в Notepad, просто сохраните ваш файл с расширением html, чтобы посмотреть его реализацию в MSIE.

Необходимое отступление о редакторах HTML

В настоящее время широко используются два типа редакторов HTML:

1.Редакторы типа "что видишь, то и получишь" (Netscape Navigator Gold, Microsoft Front Page). Пользователь не видит "внутренностей" документа, с которым он работает, точно так же, как при работе с текстовым процессором типа Microsoft Word или Word Perfect. Кстати говоря, существует довольно много конвертеров, способных преобразовывать документы, созданные в Microsoft Word или Word Perfect, в HTML-документы.

2.Редакторы собственно HTML-текстов (HotDog, Ken Nesbitt Web Editor и многие другие). В процессе работы пользователь видит внутреннее содержание HTML-файла и может изменять его либо вручную, либо вызывая команды меню для вставки определенных элементов HTML. Работа с таким редактором очень похожа на работу с интерактивной средой программирования типа Microsoft Visual Basic или Borland Delphi.

Как уже говорилось, нам для изучения HTML понадобится только текстовый редактор и браузер. Когда Вы освоите HTML в минимальной степени, Вы сами сможете подобрать себе редактор по вкусу.

Как устроен HTML-документ

Элемент HTML или гипертекстовый документ
состоит из двух частей:
•заголовка документа (HEAD)
•тела документа (BODY)

```
<HTML>
<HEAD>
Содержание заголовка
</HEAD>
<BODY>
Содержание тела документа
</BODY>
</HTML>
```

Основные классы элементов тела

Тело документа состоит из:
•Иерархических контейнеров и заставок
•Заглавий (от H1 до H6)
•Блоков (параграфы, списки, формы, таблицы, картинки и т. п.)
•Горизонтальных отчеркиваний и адресов

•Текста, разбитого на области действия стилей (подчеркивание, выделение, курсив), математические описания, графику и гипертекстовые ссылки

HTML-документ — это просто текстовый файл с расширением *.html (Unix-системы могут содержать файлы с расширением *.htmll). Вот самый простой HTML-документ:

```
<html>
   <head>
      <title>
         Пример 1
      </title>
   </head>
   <body>
      <H1>
         Привет!
      </H1>
      <P>
         Это простейший пример HTML-документа.
      </P>
      <P>
```

Этот *.html-файл может быть одновременно открыт и в Notepad, и в MSIE. Сохранив изменения в Notepad, просто сохраните ваш файл с расширением html, чтобы посмотреть его реализацию в MSIE.

```
                        </P>
   </body>
</html>
```

Для удобства чтения введены дополнительные отступы, однако в HTML это совсем не обязательно. Более того, браузеры просто игнорируют символы конца строки и множественные пробелы в HTML-файлах. Поэтому наш пример вполне мог бы выглядеть и вот так:

```
<html>
<head>
<title>Пример 1</title>
</head>
<body>
<H1>Привет!</H1>
<P>Это простейший пример HTML-документа.</P>
<P> Этот *.html-файл может быть одновременно открыт и в Aditor, и в MSIE. Сохранив изменения в Aditor, просто нажмите кнопку "обновить" в MSIE, чтобы увидеть эти изменения реализованными в HTML-документе. </P>
</body>
```

</html>

Как видно из примера, вся информация о форматировании документа сосредоточена в его фрагментах, заключенных между знаками "<" и ">". Такой фрагмент (например, <html>) называется меткой (по-английски — tag, читается "тэг").

Большинство HTML-меток — парные, то есть на каждую открывающую метку вида <tag> есть закрывающая метка вида </tag> с тем же именем, но с добавлением "/".

Метки можно вводить как большими, так и маленькими буквами. Например, метки <body>, <BODY> и <Body> будут восприняты браузером одинаково.

Многие метки, помимо имени, могут содержать атрибуты — элементы, дающие дополнительную информацию о том, как браузер должен обработать текущую метку. В нашем простейшем документе, однако, нет ни одного атрибута. Но мы обязательно встретимся с атрибутами.

Обязательные метки

<html> ... </html>

Метка <html> должна открывать HTML-документ. Аналогично, метка </html> должна завершать HTML-документ.

<head> ... </head>

Эта пара меток указывает на начало и конец заголовка документа. Помимо наименования документа (см. описание метки <title> ниже), в этот раздел может включаться множество служебной информации, о которой мы обязательно поговорим позже.

<title> ... </title>

Все, что находится между метками <title> и </title>, толкуется браузером как название документа. Браузер показывает название текущего документа в заголовке окна и печатает его в левом верхнем углу каждой страницы при выводе на принтер. Рекомендуется название не длиннее 64 символов.

<body> ... </body>

Эта пара меток указывает на начало и конец тела HTML-документа, каковое тело, собственно, и определяет содержание документа.

Основные метки простого документа

<H1> ... </H1> — <H6> ... </H6>

Метки вида <Hi> (где i — цифра от 1 до 6) описывают заголовки шести различных уровней. Заголовок первого уровня — самый крупный, шестого уровня, естественно — самый мелкий.

<P> ... </P>

Такая пара меток описывает абзац. Все, что заключено между <P> и </P>, воспринимается как один абзац.

Метки <Hi> и <P> могут содержать дополнительный атрибут ALIGN (читается "элайн", от английского "выравнивать"), например:

<H1 ALIGN=CENTER>Выравнивание заголовка по центру</H1>

или

<P ALIGN=RIGHT>Образец абзаца с выравниванием по правому краю</P>

Это атрибут задает горизонтальное выравнивание своего элемента относительно окружающего контекста. Возможные значения:

- `left`: строки текста выравниваются по левому краю.
- `center`: строки текста выравниваются по центру.
- `right`: строки текста выравниваются по правому краю.
- `justify`: строки текста выравниваются по обоим краям.

align = `left|center|right|justify`

Подытожим все, что мы знаем, с помощью примера:

<html>
<head>
<title>Пример 2</title>
</head>
<body>
<H1 ALIGN=CENTER>Привет!</H1>
<H2>Это чуть более сложный пример HTML-документа</H2>
<P>Теперь мы знаем, что абзац можно выравнивать не только влево, </P>
<P ALIGN=CENTER>но и по центру</P> <P ALIGN=RIGHT>или по правому краю.</P>
</body>
</html>

С этого момента Вы знаете достаточно, чтобы создавать простые HTML-документы самостоятельно от начала до конца. В следующей работе мы поговорим, как можно улучшить наш простой HTML-документ.

Задание

- Изучить теоретическую часть.
- Написать все примеры в редакторе и посмотреть их в браузере MS Internet Explorer.
- Создать набор заголовков и параграфов, с различным горизонтальным размещением.
- Создать основу своей странички: структуру, заголовки, текст.

ПРАКТИЧЕСКАЯ РАБОТА №3

Документ HTML. Организация текста на странице.

Цель работы: изучить элементы абзаца:

- Непарные метки
- Конец строки
- Горизонтальная линия
- &-последовательности
- Комментарии
- Форматирование шрифта
- Физические стили
- Логические стили

Познакомиться с организацией текста внутри документа:

- Ненумерованные списки
- Нумерованные списки
- Списки определений
- Вложенные списки
- Преформатированный текст
- Текст с отступом

Теоретическая часть:

Непарные метки

В этом разделе мы поговорим о метках, которые не подчиняются двум основным правилам HTML: все они непарные, а некоторые (так называемые &-последовательности) к тому же должны вводиться только маленькими буквами.

Эта метка используется, если необходимо перейти на новую строку, не прерывая абзаца. Очень удобно при публикации стихов (см. пример 1).

```
<html>
<head>
<title>Пример 1</title>
</head>
<body>
<H1>Стих</H1>
<H2>Автор неизвестен</H2>
```

```
<P>Однажды в студеную зимнюю пору<BR>
Сижу за решеткой в темнице сырой.<BR>
Гляжу - поднимается медленно в гору<BR>
Вскормленный в неволе орел молодой.</P>
<P>И шествуя важно, в спокойствии чинном,<BR>
Мой грустный товарищ, махая крылом,<BR>
В больших сапогах, в полушубке овчинном,<BR>
Кровавую пищу клюет под окном.</P>
</body> </html>
```

<HR>

Метка <HR> описывает вот такую горизонтальную линию:

Метка может дополнительно включать атрибуты SIZE (определяет толщину линии в пикселях) и/или WIDTH (определяет размах линии в процентах от ширины экрана). В примере 2 приведена небольшая коллекция горизонтальных линий.

```
<html>
<head>
<title>Пример 2</title>
</head>
<body>
<H1>Коллекция горизонтальных линий</H1>
<HR SIZE=2 WIDTH=100%><BR>
<HR SIZE=4 WIDTH=50%><BR>
<HR SIZE=8 WIDTH=25%><BR>
<HR SIZE=16 WIDTH=12%><BR>
</body>
</html>
```

&-последовательности

Поскольку символы "<" и ">" воспринимаются браузерами как начало и конец HTML-меток, возникает вопрос: а как показать эти символы на экране? В HTML это делается с помощью &-последовательностей (их еще называют символьными объектами или эскейп-последовательностями). Браузер показывает на экране символ "<", когда встречает в тексте последовательность < (по первым буквам английских слов less than — меньше, чем). Знак ">"

кодируется последовательностью > (по первым буквам английских слов greater than — больше, чем).

Символ "&" (амперсанд) кодируется последовательностью &

Двойные кавычки (") кодируются последовательностью "

Помните: точка с запятой — обязательный элемент &-последовательности. Кроме того, все буквы, составляющие последовательность, должны быть в нижнем регистре (т.е., маленькие). Использование меток типа " или & не допускается.

Вообще говоря, &-последовательности определены для всех символов из второй половины ASCII-таблицы (куда, естественно, входят и русские буквы). Дело в том, что некоторые серверы не поддерживают восьмибитную передачу данных, и поэтому могут передавать символы с ASCII-кодами выше 127 только в виде &-последовательностей.

Комментарии

Браузеры игнорируют любой текст, помещенный между <!-- и -->. Это удобно для размещения комментариев.

<!-- Это комментарий -->

Форматирование шрифта

HTML допускает два подхода к шрифтовому выделению фрагментов текста. С одной стороны, можно прямо указать, что шрифт на некотором участке текста должен быть жирным или наклонным, то есть изменить физический стиль текста. С другой стороны, можно пометить некоторый фрагмент текста как имеющий некоторый отличный от нормального логический стиль, оставив интерпретацию этого стиля браузеру. Поясним это на примерах.

Физические стили

Под физическом стилем принято понимать прямое указание браузеру на модификацию текущего шрифта. Например, все, что находится между метками <B> и </B>, будет написано жирным шрифтом. Текст между метками <I> и </I> будет написан наклонным шрифтом.

Несколько особняком стоит пара меток <TT> и </TT>. Текст, размещенный между этими метками, будет написан шрифтом, имитирующим пишущую машинку, то есть имеющим фиксированную ширину символа.

Логические стили

При использовании логических стилей автор документа не может знать заранее, что увидит на экране читатель. Разные браузеры толкуют одни и те

же метки логических стилей по-разному. Некоторые браузеры игнорируют некоторые метки вообще и показывают нормальный текст вместо выделенного логическим стилем. Вот самые распространенные логические стили.

<EM> ... </EM>

От английского emphasis — акцент.

<STRONG> ... </STRONG>

От английского strong emphasis — сильный акцент.

<CODE> ... </CODE>

Рекомендуется использовать для фрагментов исходных текстов.

<SAMP> ... </SAMP>

От английского sample — образец. Рекомендуется использовать для демонстрации образцов сообщений, выводимых на экран программами.

<KBD> ... </KBD>

От английского keyboard — клавиатура. Рекомендуется использовать для указания того, что нужно ввести с клавиатуры.

<VAR> ... </VAR>

От английского variable — переменная. Рекомендуется использовать для написания имен переменных.

Подытожим наши знания о логических и физических стилях с помощью примера 3. Заодно Вы сможете увидеть, как Ваш браузер показывает те или иные логические стили.

```
<html>
<head>
<title>Пример 3</title>
</head>
<body>
<H1>Шрифтовое выделение фрагментов текста</H1>
<P>Теперь мы знаем, что фрагменты текста можно выделять
```

<B>жирным</B> или <I>наклонным</I> шрифтом. Кроме того, можно включать в текст фрагменты с фиксированной шириной символа <TT>(имитация пишущей машинки)</TT></P>

<P>Кроме того, существует ряд логических стилей:</P>

<P><EM>EM - от английского emphasis - акцент </EM>

<STRONG>STRONG - от английского strong emphasis - сильный акцент </STRONG>

<CODE>CODE - для фрагментов исходных текстов</CODE>

<SAMP>SAMP - от английского sample - образец </SAMP>

<KBD>KBD - от английского keyboard - клавиатура</KBD>

<VAR>VAR - от английского variable - переменная </VAR></P>

</body>

</html>

Организация текста внутри документа

HTML позволяет определять внешний вид целых абзацев текста. Абзацы можно организовывать в списки, выводить их на экран в отформатированном виде, или увеличивать левое поле. Разберем все по порядку.

Ненумерованные списки: <UL> ... </UL>

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

•Иван;
•Данила;
•белая кобыла

необходим вот такой HTML-текст:

<UL>
<LI>Иван;
<LI>Данила;
<LI>белая кобыла
</UL>

Обратите внимание: у метки <LI> нет парной закрывающей метки.

Нумерованные списки: <OL> ... </OL>

Нумерованные списки устроены точно так же, как ненумерованные, только вместо символов, выделяющих новый элемент, используются цифры. Если слегка модифицировать наш предыдущий пример:

<OL>
<LI>Иван;
<LI>Данила;
<LI>белая кобыла
</OL>

получится вот такой список:

1.Иван;
2.Данила;
3.белая кобыла

Списки определений: <DL> ... </DL>

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

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

Этот фрагмент будет выведен на экран следующим образом:

HTML
Термин HTML (HyperText Markup Language) означает 'язык маркировки гипертекстов'. Первую версию HTML разработал сотрудник Европейской лаборатории физики элементарных частиц Тим Бернерс-Ли.

HTML-документ
Текстовый файл с расширением *.html (Unix-системы могут содержать файлы с расширением *.htmll).

Обратите внимание: точно так же, как метки <LI>, метки <DT> и <DD> не имеют парных закрывающих меток.

Если определяемые термины достаточно коротки, можно использовать модифицированную открывающую метку <DL COMPACT>. Например, вот такой фрагмент HTML-текста:

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

будет выведен на экран примерно так:

```
А
Первая буква алфавита
Б
Вторая буква алфавита
В
Третья буква алфавита
```

Вложенные списки

Элемент любого списка может содержать в себе целый список любого вида. Число уровней вложенности в принципе не ограничено, однако злоупотреблять вложенными списками все же не следует.

Вложенные списки очень удобны при подготовке разного рода планов и оглавлений.

Наши знания о списках можно вкратце свести в пример 4:

```
<html>
<head>
<title>Пример 4</title>
</head>
<body>
<H1>HTML поддерживает несколько видов списков </H1>
<DL>
<DT>Ненумерованные списки
<DD>Элементы ненумерованного списка выделяются специальным символом и отступом слева:
<UL>
<LI>Элемент 1
<LI>Элемент 2
<LI>Элемент 3
```

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

Преформатированный текст: <PRE> ... </PRE>

В самом начале мы говорили о том, что браузеры игнорируют множественные пробелы и символы конца строки. Из этого правила, однако, есть исключение.

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

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

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

Задание

- Изучить теоретическую часть.
- Написать все примеры в редакторе и посмотреть в MSIE.
- Ввести в свою страничку семейство линий различной толщины и длины.
- Создать набор списков, используя вложение списков:

  нумерованный

  ненумерованный

нумерованный
список определений.

- Используя различные стили выделить каким-либо образом отдельные слова, элемент списка, параграф и т.д.
- Написать в каком-либо текстовом редакторе несколько абзацев и вставить этот форматированный текст в свою страничку.
- Ввести в свою страничку примечания.

ПРАКТИЧЕСКАЯ РАБОТА №4

Изображения. Ссылки. Палитра. Заголовок.

Цель работы: изучить организацию меток и атрибутов:
• Изображения в HTML-документе;
• Связывания;
• Цветовой гаммы HTML-документа;
• Заголовка HTML-документа.

Теоретическая часть:

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

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

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

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

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

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

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

Из этого следует одно очень важное практическое соображение. Если Вы подготовили к публикации некоторую группу HTML-документов, которые ссылаются друг на друга только по имени файла и находятся в одном каталоге на Вашем компьютере, вся эта группа документов будет работать точно так же, если ее поместить в любой другой каталог на любом другом компьютере, на локальной сети или... на Интернет! Таким образом, у

Вас появляется возможность разрабатывать целые коллекции документов без подключения к Интернет, и только после окончательной готовности, подтвержденной испытаниями, помещать коллекции документов на Интернет целиком.

На практике, однако, часто бывает необходимо дать ссылку на документ, находящийся на другом сервере. Например, если Вы хотите дать ссылку на это руководство со своей странички, Вам придется ввести в свой HTML-документ примерно такой фрагмент:

```
<A HREF="http://www.yi.com/home/Chush/index.html"> всякая чушь </A>
```

При необходимости можно задать переход не просто к некоторому документу, но и к определенному месту внутри этого документа. Для этого необходимо создать в документе, к которому будет задан переход, некоторую опорную точку, или анкер. Разберем это на примере.

Допустим, что необходимо осуществить переход из файла 1.html к словам "Переход закончен" в файле 2.html (файлы находятся в одном каталоге). Прежде всего, необходимо создать вот такой анкер в файле 2.html:

```
<A NAME="AAA">Переход закончен</A>
```

Слова "Переход закончен" при этом никак не будут выделены в тексте документа.

Затем в файле 1.html (или в любом другом) можно определить переход на этот анкер:

```
<A HREF="2.html#AAA">Переход к анкеру AAA</A>
```

Кстати говоря, переход к этому анкеру можно определить и внутри самого документа 2.html — достаточно только включить в него вот такой фрагмент:

```
<A HREF="#AAA">Переход к анкеру AAA</A>
```

На практике это очень удобно при создании больших документов. В начале документа можно поместить оглавление, состоящее из ссылок на анкеры, расположенные в заголовках разделов документа.

Во избежание недоразумений рекомендуется задавать имена анкеров латинскими буквами. Следите за написанием имен анкеров: большинство браузеров отличают большие буквы от маленьких. Если имя анкера определено как AAA, ссылка на анкер aaa или AaA не выведет Вас на анкер AAA, хотя документ, скорее всего, будет загружен корректно.

Пока что мы обсуждали только ссылки на HTML-документы. Однако возможны ссылки и на другие виды ресурсов:

```
<A HREF="ftp://server/directory/file.ext">Выгрузить файл</A>
```

Такая ссылка, если ей воспользоваться, запустит протокол передачи файлов и начнет выгрузку файла file.ext, находящегося в каталоге directory на сервере server, на локальный диск пользователя.

```
<A HREF="mailto:user@mail.box">Послать письмо</A>
```

Если пользователь совершит переход по такой ссылке, у него на экране откроется окно ввода исходящего сообщения его почтовой программы. В строке To: ("Куда") окна почтовой программы будет указано user@mail.box.

Разберем все, что мы знаем о связывании, с помощью примера1.

```
<HTML>
<HEAD>
<TITLE>Пример 1</TITLE>
</HEAD>
<BODY>
<H1>Связывание </H1>
<P>С помощью ссылок можно переходить к другим файлам (например,
к
<A HREF="index.html">оглавлению
</A>).</P>
<P>Можно выгружать файлы (например,
<A HREF="ftp://yi.com/home/ChuvakhinNikolai/html-pr.doc">
руководство по HTML в формате Microsoft Word
</A>) по FTP.</P>
<P>Можно дать пользователю возможность послать почту (например,
<A HREF="mailto:am@am.tpu.ru">на кафедру ПМ
</A>).</P>
</BODY>
</HTML>
```

Изображения в HTML-документе

Встроить изображение в HTML-документ очень просто. Для этого нужно только иметь это самое изображение в формате GIF (файл с расширением *.gif) или JPEG (файл с расширением *.jpg или *.jpeg) и одну строчку в HTML-тексте.

Допустим, нам нужно включить в документ изображение, записанное в файл picture.gif, находящийся в одном каталоге с HTML-документом. Тогда строчка будет вот такая:

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

Метка <IMG SRC="[имя файла]"> может также включать атрибут ALT="[текст]", например:

<IMG SRC="picture.gif" ALT="Картинка">

Встретив такую метку, браузер покажет на экране текст Картинка и начнет загружать на его место картинку из файла picture.gif. Атрибут ALT может оказаться необходимым для старых браузеров, которые не поддерживают изображений, а также на случай, если у браузера отключена автоматическая загрузка изображений (при медленном подключении к Интернет это делается для экономии времени).

Файл, содержащий изображение, может находиться в другом каталоге или даже на другом сервере. В этом случае стоит указать его полное имя.

Разберем все, что мы знаем об изображениях, с помощью примера 2.

```
<HTML>
<HEAD>
<TITLE>Пример 2</TITLE>
</HEAD>
<BODY>
<H1>Изображения </H1>
<P>Изображение можно встроить очень просто: </P>
<P><IMG SRC="picture.gif"></P>
<P>Кроме того, изображение можно сделать "горячим", то есть
осуществлять переход при нажатии на изображение:</P>
<P><A HREF="index.html"><IMG SRC="picture.gif"></A></P>
</BODY>
</HTML>
```

Обратите внимание на вторую часть примера. Если ссылка на изображение находится между метками <A HREF="..."> и </A>, изображение фактически становится кнопкой, при нажатии на которую происходит переход по ссылке (в примере 2 переход происходит на документ index.html).

Цветовая гамма HTML-документа

Интересно, что элементы текста могут содержать HTML-метки или атрибуты, определяющие их цветовую гамму. Так в любом блочном элементе HTML-документа (параграфе, списке, таблице и др.) может присутствовать атрибут FONTCOLOR="цвет" или любая часть текста может быть выделена элементами <FONT>…</FONT> с атрибутами SIZE=… COLOR=… и т.д.

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

bgcolor

Определяет цвет фона документа.

text

Определяет цвет текста документа.

link

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

vlink

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

alink

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

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

bgcolor=#FFFFFF
Цвет фона. Насыщенность красным, зеленым и синим одинакова — FF (это шестнадцатеричное представление числа 255). Результат — белый цвет.

text=#000000
Цвет текста. Насыщенность красным, зеленым и синим одинакова — 00 (ноль). Результат — черный цвет.

link=#FF0000
Цвет гипертекстовой ссылки. Насыщенность красным — FF (255), зеленым и синим — 00 (ноль). Результат — красный цвет.

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

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

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

Заголовок HTML-документа: что в нем может быть интересного?

Заголовок HTML-документа, вообще говоря, не виден пользователю при просмотре. Однако в нем есть некоторые интересные особенности, которые стоит знать.

Заголовок HTML-документа может включать неограниченное количество так называемых META-инструкций. META-инструкция — это просто способ определить некоторую переменную путем указания ее имени (атрибут NAME) и значения (атрибут CONTENT). Вот некоторые наиболее типичные META-инструкции:

<META NAME="description" CONTENT="Это учебник для тех, кто хочет публиковать документацию любого рода на глобальной компьютерной сети Интернет">

Такая META-инструкция определяет переменную description, содержащую краткое описание документа. Многие поисковые механизмы постоянно сканируют Интернет в поисках HTML-файлов, отыскивают в них эту переменную, сохраняют ее в своих базах данных и демонстрируют ее в ответ на запросы пользователей.

<META NAME="keywords" CONTENT="Интернет, HTML, WWW, публикация, гипертекст, тег">

Такая META-инструкция определяет переменную keywords, содержащую набор ключевых слов, описывающих содержание документа. На практике поиск по ключевым словам очень удобно использовать при строительстве поискового механизма, работающего в пределах одного сервера (со стороны очень похоже, что именно такой подход был использован, например, при создании сервера технической поддержки фирмы Novell, ведущего производителя операционных систем для локальных компьютерных сетей).

Другая группа META-инструкций определяет эквиваленты команд протокола передачи гипертекстов. (Не переживайте, это не так страшно, как кажется на первый взгляд!) Разберем несколько примеров:

<META HTTP-EQUIV="Content-type"

CONTENT="text/html; charset=windows-1251">

Эта META-инструкция дает браузеру указание интерпретировать загружаемый документ как содержащий HTML-текст в кодировке Windows/1251.

<META HTTP-EQUIV="Content-type"
CONTENT="text/html; charset=koi8-r">

Эта META-инструкция абсолютно аналогична предыдущей, только в качестве кодировки указана КОИ-8.

<META HTTP-EQUIV="Refresh" CONTENT="[время]; URL=[документ]">

Такая META-инструкция дает браузеру примерно такое указание: "Если через [время] секунд после завершения загрузки этого документа пользователь не перейдет к другому документу, начать загрузку ресурса [документ]". Более конкретно это может выглядеть, к примеру, вот так:

<META HTTP-EQUIV="Refresh"
CONTENT="10; URL=http://www.server.ru/home/mypage/">

Если пользователь не предпримет никаких видимых действий в течение 10 секунд после загрузки документа, содержащего такую инструкцию, автоматически будет загружен документ http://www.server.ru/home/mypage/.

META-инструкцию Refresh можно использовать, например, если Вы перенесли некоторый документ с одного сервера на другой. Вместо копии документа на старом сервере можно оставить короткое сообщение о переносе, включающее META-инструкцию Refresh и адрес документа на новом сервере.

Если в качестве параметра [документ] подставить имя файла, содержащего звук, через [время] секунд после загрузки HTML-файла начнется загрузка и проигрывание этого звука (при условии, конечно, что браузер пользователя поддерживает формат этого звукового файла). Очень удобно для всякого рода приветственных речей.

В отличие от всех примеров, которые мы рассматривали ранее, пример 3 состоит не из одного, а из трех файлов. Используя META-инструкцию Refresh, мы создадим небольшой слайд-фильм из трех кадров (файлов e0003.html, e0003a.html и e0003b.html), которые будут циклически повторяться. Для остановки демонстрации нужно будет воспользоваться любой из гипертекстовых ссылок.

<HTML><!--файл e0003.html -->
<HEAD>
<TITLE>Пример 3</TITLE>

```
<META HTTP-EQUIV="Refresh" CONTENT="2; URL=e0003a.html">
</HEAD>
<BODY bgcolor=#FFFFFF text=#000000 link=#FF0000>
<H1>Слайд-демонстрация цветовых гамм <BR>
с помощью META-инструкции Refresh </H1>
<P>Черный текст на белом фоне </P>
<P>[<A HREF="index.html">Возврат к оглавлению</A>|
<A HREF="pr0002.html">Возврат к примеру 2</A>]</P>
</BODY>
</HTML><!--конец файла e0003.html -->

<HTML><!--файл e0003a.html -->
<HEAD>
<TITLE>Пример 3a</TITLE>
<META HTTP-EQUIV="Refresh" CONTENT="2; URL=e0003b.html">
</HEAD>
<BODY bgcolor=#000000 text=#FFFFFF link=#FF0000>
<H1>Слайд-демонстрация цветовых гамм <BR>
с помощью META-инструкции Refresh </H1>
<P>Белый текст на черном фоне </P>
<P>[<A HREF="index.html">Возврат к оглавлению</A>|
<A HREF="pr0002.html">Возврат к примеру 2</A>]</P>
</BODY>
</HTML><!--конец файла e0003a.html -->

<HTML><!--файл e0003b.html -->
<HEAD>
<TITLE>Пример 3b</TITLE>
<META HTTP-EQUIV="Refresh" CONTENT="2; URL=e0003.html">
</HEAD>
<BODY bgcolor=#C0C0C0 text=#0000FF link=#FF0000>
<H1>Слайд-демонстрация цветовых гамм <BR>
с помощью META-инструкции Refresh </H1>
<P>Синий текст на сером фоне </P>
<P>[<A HREF="index.html">Возврат к оглавлению</A>|
<A HREF="pr0002.html">Возврат к примеру 2</A>]</P>
</BODY>
</HTML><!--конец файла e0003b.html -->
```

Задание

- Изучить теоретическую часть.
- Написать все примеры в редакторе и посмотреть в MSIE.
- Ввести в свою страничку ссылки в пределах одной страницы, ссылку на другой документ и ссылку на закладку в другом документе.

- Ввести в свою страничку изображения.
- Сделать некоторое изображение ссылкой, а другое закладкой.
- Создать фон странички.
- Задать цветовое оформление текста и ссылок.
- Выделить цветом параграф, заголовок, часть текста.
- Ввести в заголовок своей странички META-информацию:
    - Автор
    - Дата создания
    - Версия
    - Краткое описание
    - Ключевые слова
    - Кодировку
    - Refresh

ПРАКТИЧЕСКАЯ РАБОТА №5

Таблицы. Фреймы.

Цель работы: познакомиться с организацией непоследовательного размещения информации на странице:

- Таблицы;
- Фреймы.

Теоретическая часть:

Для чего нужны таблицы?

На этот вопрос есть очевидный ответ: таблицы нужны для представления информации в табличном виде. Есть, однако, и менее очевидные ответы.

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

Кроме того, таблица, состоящая из одной ячейки, может очень эффектно выделить фрагмент текста, на который Вы хотите обратить внимание читателя.

Как устроена таблица

В устройстве таблицы легче всего разобраться на простом примере.

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

```
<TR>          <!--Это начало второй строки-->
<TD>          <!--Это начало первой ячейки-->
Вторая строка, первая колонка
</TD>         <!--Это конец первой ячейки-->
<TD>          <!--Это начало второй ячейки-->
Вторая строка, вторая колонка
</TD>         <!--Это конец второй ячейки-->
</TR>         <!--Это конец второй строки-->
</TABLE>      <!--Это конец таблицы-->
</BODY>
</HTML>
```

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

ALIGN

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

WIDTH

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

BORDER

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

CELLSPACING

Устанавливает расстояние между рамками ячеек таблицы в пикселях (например, CELLSPACING=2).

CELLPADDING

Устанавливает расстояние между рамкой ячейки и текстом в пикселях (например, CELLPADDING=10).

Таблица может иметь заголовок (<CAPTION> ... </CAPTION>), хотя заголовок не является обязательным. Метка <CAPTION> может включать атрибут ALIGN. Допустимые значения: <CAPTION ALIGN=TOP> (заголовок помещается над таблицей) и <CAPTION ALIGN=BOTTOM> (заголовок помещается под таблицей).

Каждая строка таблицы начинается с метки <TR> и заканчивается меткой </TR>. Метка <TR> может включать следующие атрибуты:

ALIGN

Устанавливает выравнивание текста в ячейках строки. Допустимые значения: ALIGN=LEFT (выравнивание влево), ALIGN=CENTER (выравнивание по центру), ALIGN=RIGHT (выравнивание вправо).

VALIGN

Устанавливает вертикальное выравнивание текста в ячейках строки. Допустимые значения: VALIGN=TOP (выравнивание по верхнему краю), VALIGN=MIDDLE (выравнивание по центру), VALIGN=BOTTOM (выравнивание по нижнему краю).

Каждая ячейка таблицы начинается с метки <TD> и заканчивается меткой </TD>. Метка <TD> может включать следующие атрибуты:

NOWRAP

Присутствие этого атрибута означает, что содержимое ячейки должно быть показано в одну строку.

COLSPAN

Устанавливает "размах" ячейки по горизонтали. Например, COLSPAN=3 означает, что ячейка простирается на три колонки.

ROWSPAN

Устанавливает "размах" ячейки по вертикали. Например, ROWSPAN=2 означает, что ячейка занимает две строки.

ALIGN

Устанавливает выравнивание текста в ячейке. Допустимые значения: ALIGN=LEFT (выравнивание влево), ALIGN=CENTER (выравнивание по центру), ALIGN=RIGHT (выравнивание вправо).

VALIGN

Устанавливает вертикальное выравнивание текста в ячейке. Допустимые значения: VALIGN=TOP (выравнивание по верхнему краю), VALIGN=MIDDLE (выравнивание по центру), VALIGN=BOTTOM (выравнивание по нижнему краю).

WIDTH

Устанавливает ширину ячейки в пикселях (например, WIDTH=200).

HEIGHT

Устанавливает высоту ячейки в пикселях (например, HEIGHT=40).

Если ячейка таблицы пуста, вокруг нее не рисуется рамка. Если ячейка пуста, а рамка нужна, в ячейку можно ввести символьный объект   (non-breaking space — не разрывающий пробел). Ячейка по-прежнему будет пустой, а рамка вокруг нее будет.

Ячейка таблицы может содержать в себе: текст (неформатированный или форматированный: абзац, список любого вида, преформатированный текст и т.д.), изображение, ссылку, форму или может содержать в себе другую таблицу.

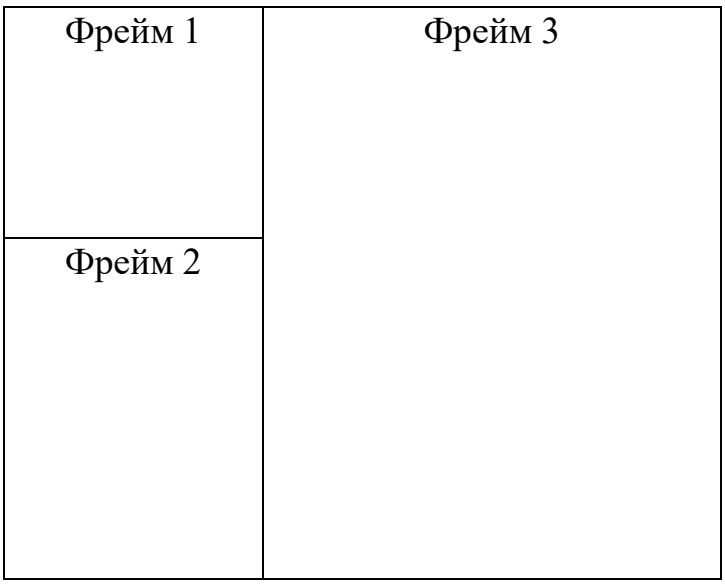
Фреймы

Фреймы в HTML позволяют авторам представлять документы в нескольких разделах, которые могут быть независимыми или вложенными окнами. Это обеспечивает дизайнерам способ оставлять некоторую информацию видимой, в то время как другая информация прокручивается или заменяется. Например, в одном окне в одном фрейме может отображаться статический баннер, во втором навигационное меню, а в третьем - сам документ, который можно прокручивать или переходить к другому с помощью навигации во втором фрейме.

Вот простой документ с использованием фреймов:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN">
<HTML>
<HEAD>
<TITLE>Простой документ с фреймами</TITLE>
</HEAD>
<FRAMESET cols="20%, 80%">
  <FRAMESET rows="100, 200">
     <FRAME src="contents_of_frame1.html">
     <FRAME src="contents_of_frame2.gif">
  </FRAMESET>
  <FRAME src="contents_of_frame3.html">
  <NOFRAMES>
     <P>В этом документе содержится:
     <UL>
       <LI><A href="contents_of_frame1.html"> Миленький текстик </A>
       <LI><IMG src="contents_of_frame2.gif" alt="Симпатичная
картинка">
       <LI><A href="contents_of_frame3.html"> Еще славный текстик</A>
     </UL>
  </NOFRAMES>
</FRAMESET>
</HTML>
```

это может привести примерно к следующей структуре экрана:



Если агент пользователя не может представлять фреймы или сконфигурирован так, чтобы не делать этого, он должен генерировать содержимое элемента NOFRAMES.

Расположение фреймов

Документ HTML, в котором описывается компоновка фреймов (называемый *документом с фреймами*), выглядит не так, как документ HTML без фреймов. Стандартный документ имеет один раздел HEAD и один раздел BODY. Документ с фреймами имеет раздел HEAD и раздел FRAMESET, который заменяет раздел BODY.

В разделе FRAMESET задается расположение фреймов в основном окне агента пользователя. Кроме того, в разделе FRAMESET может присутствовать элемент NOFRAMES с альтернативным содержимым для агентов пользователей, не поддерживающих фреймы или сконфигурированных так, чтобы их не показывать.

Элементы, обычно помещаемые в раздел BODY, не должны присутствовать до первого элемента FRAMESET, иначе элемент FRAMESET будет игнорироваться.

Элемент FRAMESET определяет макет основного окна пользователя в виде прямоугольных пространств.

Элемент FRAMESET имеет следующие атрибуты:

rows =

Этот атрибут определяет расположение горизонтальных фреймов. Это разделенный запятыми список пикселей, процентов и относительных длин. По умолчанию используется 100%, что означат одну строку.

cols =

Этот атрибут определяет расположение вертикальных фреймов. Это разделенный запятыми список пикселей, процентов и относительных длин. По умолчанию используется 100%, что означат один столбец.

Если атрибут ROWS не установлен, каждый столбец занимает всю длину страницы. Если атрибут COLS не установлен, каждая строка занимает всю ширину страницы. Если не установлен ни один из этих атрибутов, фрейм занимает всю страницу.

Фреймы создаются в направлении слева направо для столбцов и сверху вниз для строк. Если указаны оба атрибута, разделы окон создаются слева направо в верхней строке, слева направо во второй строке и т.д.

Вложенные наборы фреймов

Число уровней вложенности фреймов не ограничено.

В следующем примере внешний элемент FRAMESET разделяет доступное пространство на три равных столбца. Внутренний элемент FRAMESET разделяет вторую область на две строки неравной высоты.

```
<FRAMESET cols="33%, 33%, 34%">
   ...содержимое первого фрейма...
   <FRAMESET rows="40%, 50%">
     ...содержимое второго фрейма, первая строка...
     ...содержимое второго фрейма, вторая строка...
   </FRAMESET>
   ...содержимое третьего фрейма...
</FRAMESET>
```

Элемент FRAME определяет содержимое и вид одного фрейма.

Определения атрибутов:

name =

Назначает имя текущему фрейму. Это имя может использоваться в качестве цели в последующих ссылках.

longdesc =

Ссылка на длинное описание фрейма. Это объявление должно дополнять краткое описание, задаваемое атрибутом title, и может быть особенно полезно для невизуальных агентов пользователей.

<FRAME src="ostrich.gif" longdesc="ostrich-desc.html">

src =

Определяет местонахождение начального содержимого фрейма.

Содержимое фрейма и его определение не должны находиться в одном документе.

noresize

Если этот атрибут присутствует, он сообщает агенту пользователя, что размеры фрейма изменять нельзя.

scrolling = `auto|yes|no`

Этот атрибут задает информацию о прокрутке фрейма. Возможные значения:

- `auto`: При необходимости предоставлять возможности прокрутки. Это значение используется по умолчанию.
- `yes`: Всегда предоставлять возможности прокрутки.
- `no`: Не предоставлять возможности прокрутки.

frameborder = `1|0`

Этот атрибут предоставляет агенту пользователя информацию о границе фрейма. Возможные значения:

- `1`: Агент пользователя должен изобразить разделитель между этим фреймам и всеми прилежащими фреймами. Это значение используется по умолчанию.
- `0`: Агент пользователя не должен отображать разделитель. Обратите внимание, что разделители могут все равно отображаться, если они заданы в других фреймах.

marginwidth = *пиксели*

Этот атрибут задает пространство, оставляемое во фрейме в качестве левого и правого полей. Значение должно превышать один пиксел. Значение по умолчанию зависит от агента пользователя.

marginheight = *пиксели*

Этот атрибут определяет верхнее и нижнее поля в фрейме. Значение должно превышать один пиксель. Значение по умолчанию зависит от агента пользователя.

Атрибуты, определяемые в другом месте:

target = name_frame

Задает имя фрейма, в котором должен открываться документ.

Назначая фрейму имя с помощью атрибута NAME, авторы могут ссылаться на него как на "target" для ссылок, определяемый другими элементами. Атрибут TARGET может устанавливаться для элементов, создающих ссылки (A, LINK), навигационных карт (AREA) и форм (FORM).

...

<FRAMESET ROWS="20%,*" COLS="100,400" border="1" bordercolor="blue">

<frame name="one" src="one.html" noresize scrolling="no">

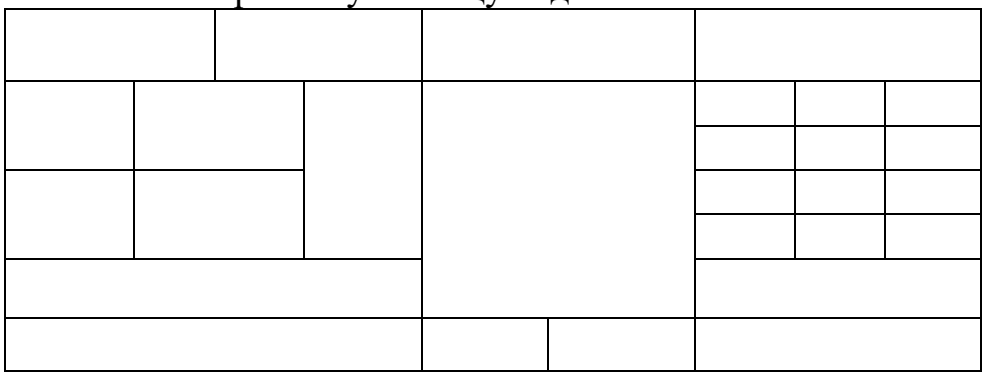
```
<frame name="two" src="two.html"  scrolling="yes">
<FRAMESET>
...

...
<head>
<title> one.html </title>
</head>
...
<a href="tree.html" target="two">открыть tree.html во втором фрейме</a>
...
```

Примечание. *Определение набора фреймов никогда не изменяется, но содержимое одного из фреймов может изменяться. При изменении исходного содержимого одного из фреймов определение набора фреймов более не отражает текущего состояния фреймов.*

Задание

- Изучить теоретическую часть.
- Написать все примеры в редакторе и посмотреть в MSIE.
- Ввести в свою страничку таблицу вида:



- Разместить в ячейках таблицы текст, изображения, списки.
- Сделать некоторые ячейки ссылкой, а другие закладкой.
- Задать цветовое оформление таблицы.
- Создать документ с определением фреймов: фиксированного (меню) и динамического.
- Создать фрейм меню со ссылками на другие документы с отображением в динамическом фрейме.

ПРАКТИЧЕСКАЯ РАБОТА №6

Формы.

Цель работы: выяснить:
•Для чего нужны формы?
•Как устроена форма
•Как форма собирает данные
•Как отправить форму почтой

Теоретическая часть:

Форма — это инструмент, с помощью которого HTML-документ может послать некоторую информацию в некоторую заранее определенную точку внешнего мира, где информация будет некоторым образом обработана.

Рассказать о формах в работе, посвященной HTML, достаточно трудно. Причина очень простая: создать форму гораздо проще, чем ту "точку внешнего мира", в которую форма будет посылать информацию. В качестве такой "точки" в большинстве случаев выступает программа, написанная на Перл или Си. Программы, обрабатывающие данные, переданные формами, часто называют CGI-скриптами. Сокращение CGI (Common Gateways Interface) означает "общепринятый интерфейс шлюзов". Написание CGI-скриптов в большинстве случаев требует хорошего знания соответствующего языка программирования и возможностей операционной системы Unix.

Также определенное распространение имеют языки JAVA и Vbscript, инструкции (скрипты) которых можно встраивать прямо в HTML-документы.

Формы передают информацию программам-обработчикам в виде пар [имя переменной]=[значение переменной]. Имена переменных следует задавать латинскими буквами. Значения переменных воспринимаются обработчиками как строки, даже если они содержат только цифры.

Как устроена форма

Форма открывается меткой <FORM> и заканчивается меткой </FORM>. 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.

Метка <FORM> может содержать три атрибута, один из которых является обязательным. Вот эти атрибуты:

ACTION

URL сервера запросов, куда будет отослано содержание формы после подтверждения. Если это поле отсутствует, будет использован URL текущего документа.

METHOD

HTTP/1.0 метод используемый для посылки содержания заполненной формы на сервер. Этот метод зависит от того, как работает конкретный сервер запросов. Настоятельно рекомендуется использование метода POST. Возможные варианты следующие:

•GET - это метод по умолчанию, который приводит к добавлению содержимого заполненной формы к URL, как и в нормальном запросе.

•POST при использовании этого метода содержимое заполненной формы пересылается не как часть URL, а как содержимое тела запроса.

ENCTYPE

задает тип кодирования содержимого заполненной формы. Этот атрибут действует только когда используется метод POST и даже в этом случае имеет только одно возможное значение (которое является значением по умолчанию)- application/x-www-form-urlencoded.

Внутри FORM оператора может находиться все, что угодно, кроме другого оператора FORM. Согласно спецификации, для задания интерфейсных элементов внутри оператора FORM используются тэги INPUT, SELECT, и TEXTAREA.

Простейшая форма

Для того, чтобы запустить процесс передачи данных из формы обработчику, нужен какой-то орган управления. Создать такой орган управления очень просто:

<INPUT TYPE=submit>

Встретив такую строчку внутри формы, браузер нарисует на экране кнопку с надписью Submit (читается "сабмит" с ударением на втором слоге, от английского "подавать"), при нажатии на которую все имеющиеся в форме данные будут переданы обработчику, определенному в метке <FORM>.

Надпись на кнопке можно задать такую, какая нравится, путем введения атрибута VALUE="[Надпись]" (читается "вэлью" с ударением на первом слоге, от английского "значение"), например:

<INPUT TYPE=submit VALUE="Поехали!">

Теперь мы знаем достаточно для того, чтобы написать простейшую форму (пример 1). Она не будет собирать никаких данных, а просто откроет страничку Index.html, как по ссылке.

<HTML>

```
<HEAD>
<TITLE>Пример 1</TITLE>
</HEAD>
<H1>Простейшая форма </H1>
<FORM ACTION="Index.html"> <!--Это начало формы-->
<INPUT TYPE=submit VALUE="Поехали...">
</FORM> <!--Это конец формы-->
</BODY>
</HTML>
```

Надпись, нанесенную на кнопку, можно при необходимости передать обработчику путем введения в определение кнопки атрибута NAME=[имя] (читается "нэйм", от английского "имя"), например:

```
<INPUT TYPE=submit NAME=button VALUE="Поехали!">
```

При нажатии на такую кнопку обработчик вместе со всеми остальными данными получит и переменную button со значением Поехали!.

В форме может быть несколько кнопок типа submit с различными именами и/или значениями. Обработчик, таким образом, может действовать по-разному в зависимости от того, какую именно кнопку submit нажал пользователь.

Как форма собирает данные

Существуют и другие типы элементов <INPUT>. 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.

Основные типы элементов <INPUT>:

TYPE=text

Определяет окно для ввода строки текста. Может содержать дополнительные атрибуты SIZE=[число] (ширина окна ввода в символах) и MAXLENGTH=[число] (максимально допустимая длина вводимой строки в символах). Пример:

```
<INPUT TYPE=text SIZE=20 NAME=user VALUE="Иван">
```

Определяет окно шириной 20 символов для ввода текста. По умолчанию в окне находится текст Иван, который пользователь может редактировать. Отредактированный (или неотредактированный) текст передается обработчику в переменной user.

TYPE=password

Определяет окно для ввода пароля. Абсолютно аналогичен типу text, только вместо символов вводимого текста показывает на экране звездочки (*). Пример:

<INPUT TYPE=password NAME=pw SIZE=20 MAXLENGTH=10>

Определяет окно шириной 20 символов для ввода пароля. Максимально допустимая длина пароля — 10 символов. Введенный пароль передается обработчику в переменной pw.

TYPE=radio

Определяет радиокнопку. Может содержать дополнительный атрибут checked (показывает, что кнопка помечена). В группе радиокнопок с одинаковыми именами может быть только одна помеченная радиокнопка. Пример:

<INPUT TYPE=radio NAME=modem VALUE="9600" checked> 9600 бит/с
<INPUT TYPE=radio NAME=modem VALUE="14400"> 14400 бит/с
<INPUT TYPE=radio NAME=modem VALUE="28800"> 28800 бит/с

Определяет группу из трех радиокнопок, подписанных 9600 бит/с, 14400 бит/с и 28800 бит/с. Первоначально помечена первая из кнопок. Если пользователь не отметит другую кнопку, обработчику будет передана переменная modem со значением 9600. Если пользователь отметит другую кнопку, обработчику будет передана переменная modem со значением 14400 или 28800.

TYPE=checkbox

Определяет квадрат, в котором можно сделать пометку. Может содержать дополнительный атрибут checked (показывает, что квадрат помечен). В отличие от радиокнопок, в группе квадратов с одинаковыми именами может быть несколько помеченных квадратов. Пример:

<INPUT TYPE=checkbox NAME=comp VALUE="PC"> Персональные компьютеры
<INPUT TYPE=checkbox NAME=comp VALUE="WS" checked> Рабочие станции
<INPUT TYPE=checkbox NAME=comp VALUE="LAN"> Серверы локальных сетей

<INPUT TYPE=checkbox NAME=comp VALUE="IS" checked> Серверы Интернет

Определяет группу из четырех квадратов. Первоначально помечены второй и четвертый квадраты. Если пользователь не произведет изменений, обработчику будут переданы две переменные: comp=WS и comp=IS.

TYPE=hidden

Определяет скрытый элемент данных, который не виден пользователю при заполнении формы и передается обработчику без изменений. Такой элемент иногда полезно иметь в форме, которая время от времени подвергается переработке, чтобы обработчик мог знать, с какой версией формы он имеет дело. Другие возможные варианты использования Вы вполне можете придумать сами. Пример:

<INPUT TYPE=hidden NAME=version VALUE="1.1">

Определяет скрытую переменную version, которая передается обработчику со значением 1.1.

TYPE=reset

Определяет кнопку, при нажатии на которую форма возвращается в исходное состояние. Поскольку при использовании этой кнопки данные обработчику не передаются, кнопка типа reset может и не иметь атрибута name. Пример:

<INPUT TYPE=reset VALUE="Очистить поля формы">

Определяет кнопку Очистить поля формы, при нажатии на которую форма возвращается в исходное состояние.

Помимо элементов <INPUT>, формы могут содержать меню <SELECT> и поля для ввода текста <TEXTAREA>.

Меню <SELECT> из n элементов выглядит примерно так:

<SELECT NAME="[имя]">
<OPTION VALUE="[значение 1]">[текст 1]
<OPTION VALUE="[значение 2]">[текст 2]
...
<OPTION VALUE="[значение n]">[текст n]
</SELECT>

Как Вы видите, меню начинается с метки <SELECT> и заканчивается меткой </SELECT>. Метка <SELECT> содержит обязательный атрибут NAME, определяющий имя переменной, которую генерирует меню.

Метка <SELECT> может также содержать атрибут MULTIPLE, присутствие которого показывает, что из меню можно выбрать несколько элементов. Большинство браузеров показывают меню <SELECT MULTIPLE> в виде окна, в котором находятся элементы меню (высоту окна в строках можно задать атрибутом SIZE=[число]). Меню <SELECT> в большинстве случаев показывается в виде выпадающего меню.

Метка <OPTION> определяет элемент меню. Обязательный атрибут VALUE устанавливает значение, которое будет передано обработчику, если выбран этот элемент меню. Метка <OPTION> может включать атрибут checked, показывающий, что данный элемент отмечен по умолчанию.

Разберем небольшой пример.

```
<SELECT NAME="selection">
<OPTION VALUE="option1" checked>Вариант 1
<OPTION VALUE="option2">Вариант 2
<OPTION VALUE="option3">Вариант 3
</SELECT>
```

Такой фрагмент определяет меню из трех элементов: Вариант 1, Вариант 2 и Вариант 3. По умолчанию выбран элемент Вариант 1. Обработчику будет передана переменная selection значение которой может быть option1 (по умолчанию), option2 или option3.

После всего, что мы уже узнали, элемент <TEXTAREA> может показаться совсем простым. Например:

```
<TEXTAREA NAME=address ROWS=5 COLS=50>
А здесь - Ваш адрес...
</TEXTAREA>
```

Все атрибуты обязательны. Атрибут NAME определяет имя, под которым содержимое окна будет передано обработчику (в примере — address). Атрибут ROWS устанавливает высоту окна в строках (в примере — 5). Атрибут COLS устанавливает ширину окна в символах (в примере — 50).

Текст, размещенный между метками <TEXTAREA> и </TEXTAREA>, представляет собой содержимое окна по умолчанию. Пользователь может его отредактировать или просто стереть.

Важно знать, что русские буквы в окне <TEXTAREA> при передаче обработчику могут быть конвертированы в соответствующие им символьные объекты.

Для демонстрации использования форм я написал небольшую программу на PHP, которая находится по адресу:

http://206.31.82.215/hp/nc/fd-win.pht

Исходные данные в эту программу передаст форма, описанная в примере 2:

```
<HTML>
<HEAD>
<TITLE>Пример 2</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>
```

Заполняйте форму, жмите на кнопку и смотрите, что будет...

Как отправить форму почтой

Все это прекрасно, скажут скептики, но на кой ляд нужны формы людям, которым их нечем обработать? Отчасти это верно, но только отчасти.

HTML предоставляет в Ваше распоряжение довольно мощный механизм пересылки содержимого форм по электронной почте. Вот как это выглядит на практике.

Допустим, что мы слегка изменили Пример 2. Вместо строки

```
<FORM ACTION="http://206.31.82.215/hp/nc/fd-win.pht" METHOD=post>
```

мы ввели строку

```
<FORM ACTION="mailto:user@mail.box" ENCTYPE=text/plain>
```

Обратите внимание, что мы изменили алгоритм кодирования на text/plain, то есть фактически выключили кодирование вообще.

Предположим теперь, что пользователь указал, имя Иван, фамилию Петров, мужской пол и возраст 22 года. Теперь вопрос: что произойдет, если пользователь нажмет на кнопку Запустить обработчик?

Ответ прост. На адрес user@mail.box электронной почтой автоматически будет отправлено сообщение следующего содержания:

fn=Иван
ln=Петров
gender=male
age=22

К сожалению, не все пользователи смогут воспользоваться такой формой. Дело в том, что всю работу по составлению сообщения и запуску почтовой программы для его отправки фактически берет на себя браузер пользователя. Это значит, что конфигурация доступа пользователя к Интернет должна обеспечивать одновременное функционирование протокола передачи гипертекстов и протокола доставки исходящей почты. Такое возможно не всегда. Тем не менее, даже если это невозможно, ничего смертельного не случится. Браузер просто выдаст сообщение об ошибке.

Процедуры обработки событий формы на VBScript

Что такое эти процедуры? Процедура это общее название функций или подпрограмм. Имеет синтаксис:

Sub name_событие ..... end sub.

name - это имя процедуры, например, мы написали, что кнопка "загадать число" называется "chislo", значит подпрограмма sub chislo_onclick выполнится при нажатии на эту кнопку.

событие - то, что может "случиться" в окне броузера. Например "onclick" означает, что процедура выполнится, при нажатии на кнопку, предварительно заданную в форме.

Существует несколько видов событий. Из самых распространенных можно отметить: window_onload - запускается при полной загрузке документа:

```
<Script language="VBscript">
<!-- Sub window_onload
Alert "Добро пожаловать на мою домашнюю страницу!"
end sub -->
</Script>
```

При загрузке документа появится окошко с данной надписью и кнопкой ОК

Для того чтобы браузер мог различать команды VBScript, нужно все операторы VBScript на HTML-страницах обрамлять тегами <SCRIPT> и </SCRIPT>. Первый из них используется в паре с атрибутом LANGUAGE для определения языка создания сценария. В нашем случае - Visual Basic Script (хотя может быть и JavaScript). Значением для этого языка является "VBScript":

<HTML>

```
<HEAD><TITLE>Пример странички с фрагментом на VBScript</TITLE>
<SCRIPT LANGUAGE="VBScript">
<!-
Sub Button1_OnClick
    MsgBox "VBScript - Rulez Forever!"
End Sub
-></SCRIPT>
</HEAD>
<BODY>
<H3>Обычная первая страничка</H3><HR>
<FORM><INPUT NAME="Button1" TYPE="BUTTON" VALUE="Click Here"></FORM>
</BODY>
</HTML>
```

Тег <SCRIPT> имеет завершающую часть - </SCRIPT>. Всегда употребляйте их парой! Сценарий в нашем примере помещается в специальные скобки <!-...->, которые в языке HTML обозначают комментарий. Это делается для того, чтобы старые браузеры, которые не умеют работать со скриптовыми языками, случайно не отобразили сценарий на экране. Для них он - просто комментарий.

Задание

- Изучить теоретическую часть.
- Написать все примеры в редакторе и посмотреть в MSIE.
- Ввести в свою страничку форму и наполнить ее различными управляющими элементами (посмотреть, как выглядят, и проверить, как работают, все типы элемента INPUT, меню <SELECT> и поля для ввода текста <TEXTAREA>).
- Послать данные формы по текущему адресу, по адресу другой вашей страницы, по электронной почте.

ПРАКТИЧЕСКАЯ РАБОТА №7
СОЗДАНИЕ ДИНАМИЧЕСКИХ БАННЕРОВ

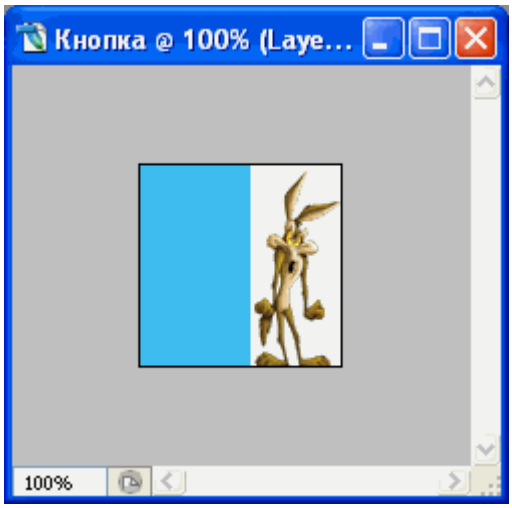
Цель занятия

1. Изучить технологию создания динамического баннера.
2. Изучить технологию покадровой анимации.
3. Специальные эффекты.

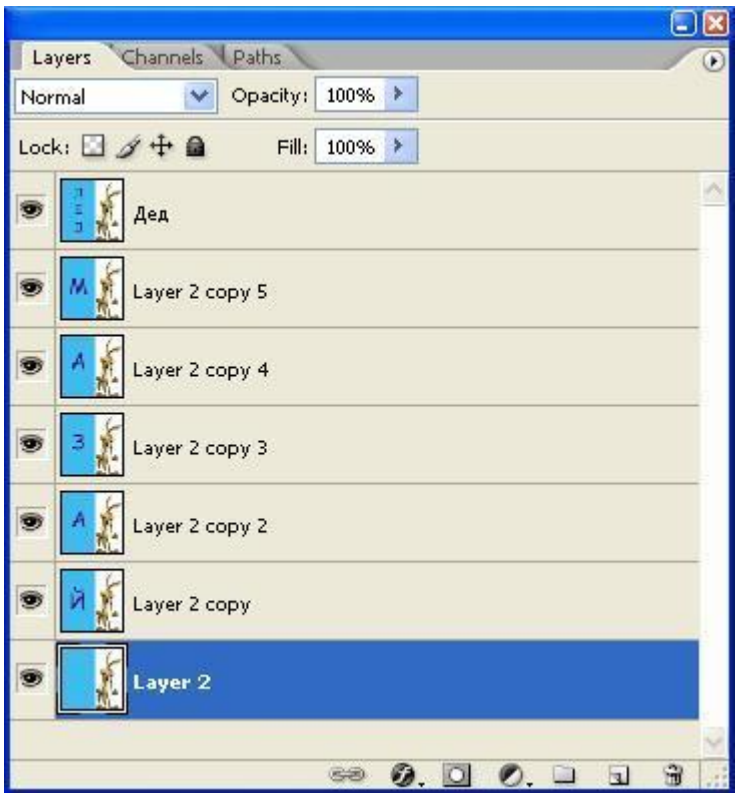
Программное обеспечение
Adobe Photoshop
Photoshop онлайн http://editor.0lik.ru/
Gimp (свободный графический редактор)
Ссылки в интернете на свободные ресурсы
Бесплатные анимированные флэш-баннеры с оригинальными эффектами
http://www.123-banner.com/

Задание 1. Создание рекламного баннера размером 100x100 в формате GIF.

Для изготовления этого баннера будем использовать только заливку и надписи.
Создайте слой с заливкой и небольшим рисунком в уголке. Рисунок любой



Затем, в каждый слой вносите элемент, который будет меняться. Например, текст, который разбит на буквы.



Переключитесь в Adobe ImageReady и задайте анимацию со временем показа кадра 0.5 секунды

Сохраните баннер оптимизированным

Задание 3. Создание рекламных баннеров на заданную тему.

Необходимо создать серию имиджевых баннеров (размером 468x60, 100x100) единой тематики, разработав слоган, визуал, цветовую гамму, текст.

Обязательные элементы:

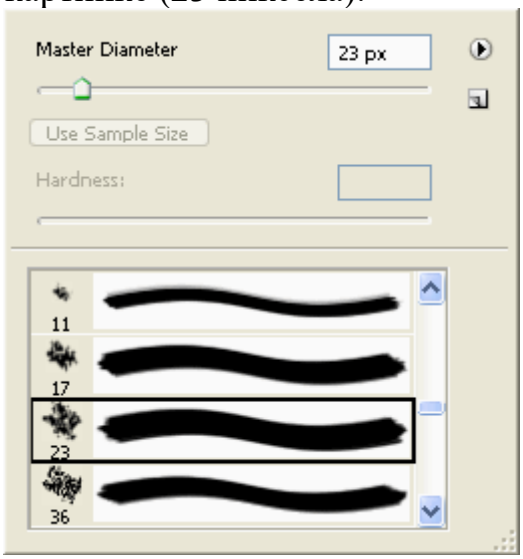
Единая цветовая гамма;

Рекламный слоган;

Задание 2 . Создание собственного аватара

Создаем новый документ. (200x200, цвет - белый)

Берем кисть (brush tool). Выбираем из списка кисть, похожую на мазки и изображённую на картинке (23 пиксела).



Создаем новый слой (Слой – Новый - Слой) и рисуем кистью как показано ниже.

Вставляем фото. Мы используем изображение Моны Лизы.
Если он не подходит по размерам, используем инструмент Свободное трансформирование

Жмем Ctrl+Alt+G, или Layer > Create clipping mask

Обратите внимание на палитру Слои. Она выглядит следующим образом:

Выделяем Слой1 и Слой2 и объединяем оба слоя (Слой – Объединить слои). Дублируем получившийся слой (Ctrl+J).
Затем берем ластик (Eraser tool) с такими же настройками кисти как и раньше. Дублируем слой и делаем невидимым предыдущий. Стираем нижний штрих как показано ниже.

Таким же образом несколько раз дублируем слои и стираем штрихи.

Переходим в Image Ready. Открываем окно анимации. Для создания первого кадра спрячем все слои. Далее нажимаем на панели значок Дублировать кадр и делаем видимым второй слой на палитре. Продолжаем дублировать фреймы, постепенно делая видимыми спрятанные слои.

Жмем Ctrl+Alt+Shift+S (File > Save optimized As), чтобы сохранить анимацию.

Самостоятельная работа

1. Обзор ПО для решения профессиональных задач.
2. Обзор профессиональных пакетов.
3. Основные ресурсы Интернет.

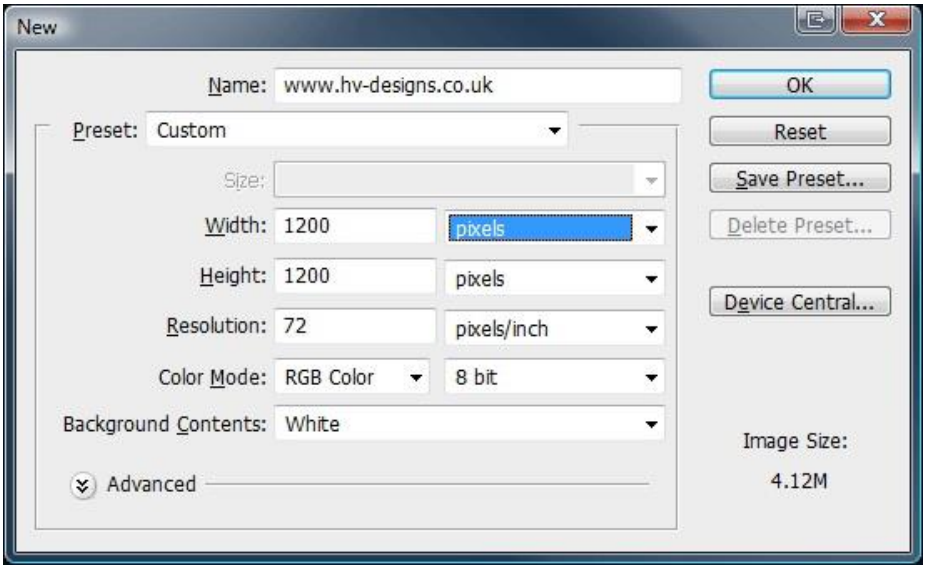
ПРАКТИЧЕСКАЯ РАБОТА №8
СОЗДАНИЕ МАКЕТА САЙТА.

Цель занятия

1. Изучить технологию создания макета сайта.

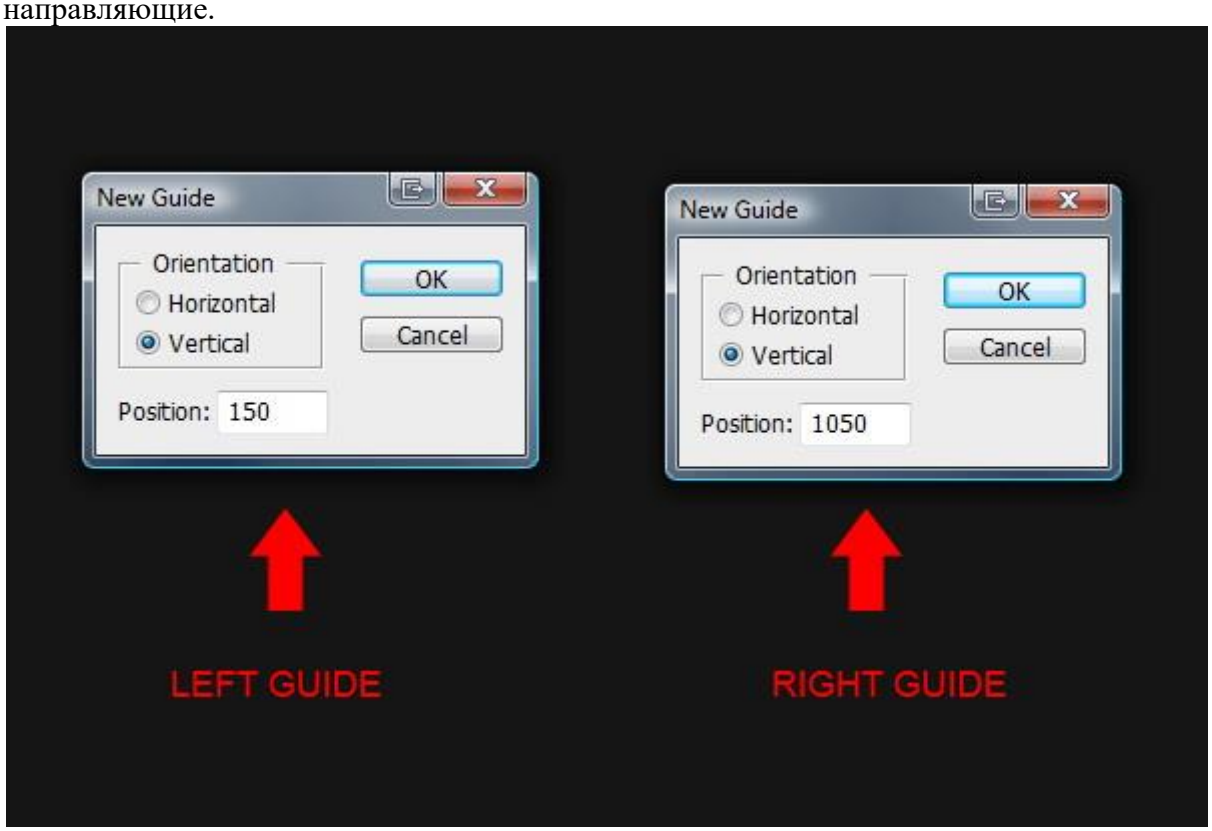
Программное обеспечение
Adobe Photoshop
Photoshop онлайн http://editor.0lik.ru/
Gimp (свободный графический редактор)

1. Создайте новый документ:

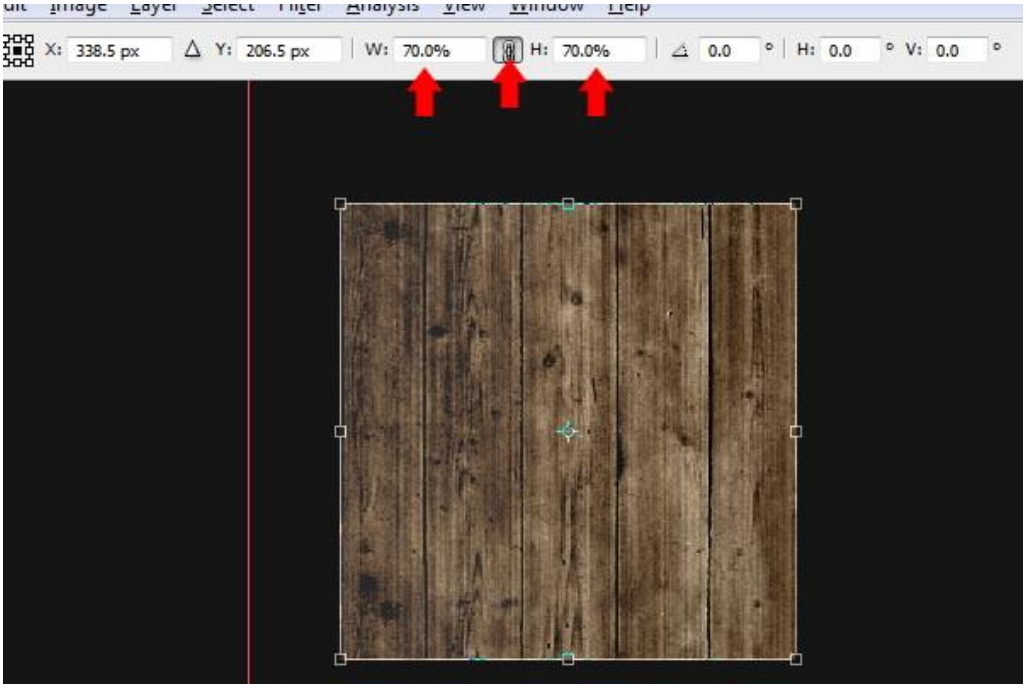


2. Установите цвет заливки - #151515 и залейте фон, используя инструмент Заливка (Paint Bucket Tool). Нам понадобятся две направляющие, чтобы область контента была шириной 900 пикселей.

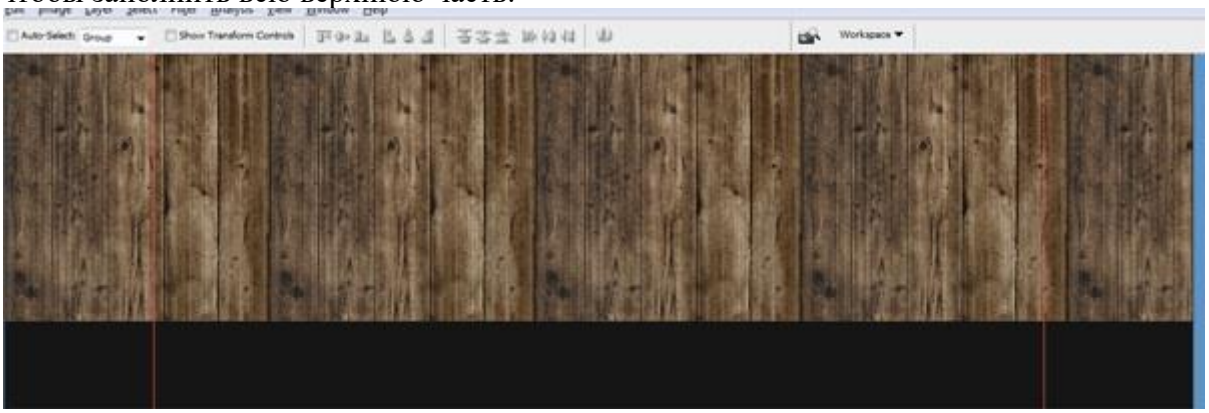
Перейдите в меню Просмотр»Новая направляющая (View > New Guide) и создайте там две направляющие.



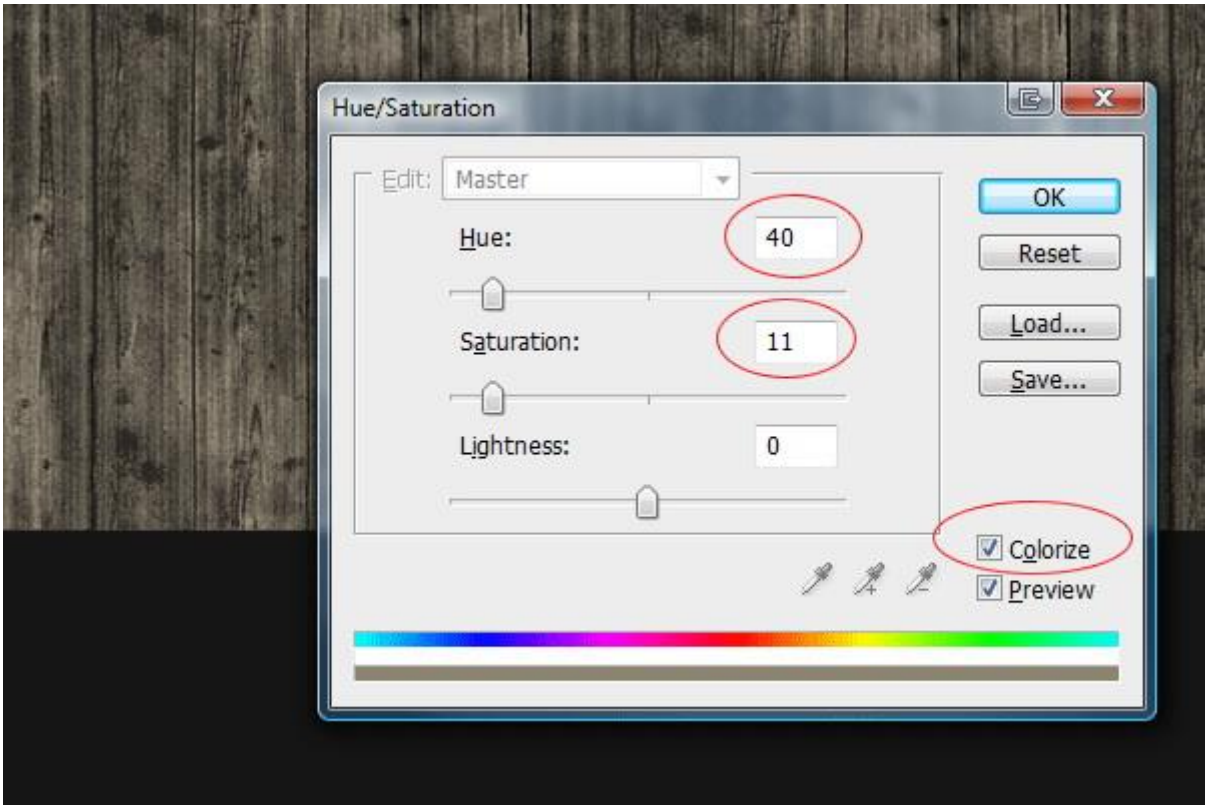
3. Для заголовка мы воспользуемся деревянной текстурой, которую вы можете найти здесь. Откройте её и вставьте в наш документ, уменьшите её до 30%. Для этого нужно перейти в режим свободного трансформирования (Ctrl+T). Используйте настройки, показанные ниже.



4. Поместите изменённую текстуру в левый верхний угол, сделайте несколько копий, чтобы заполнить всю верхнюю часть.



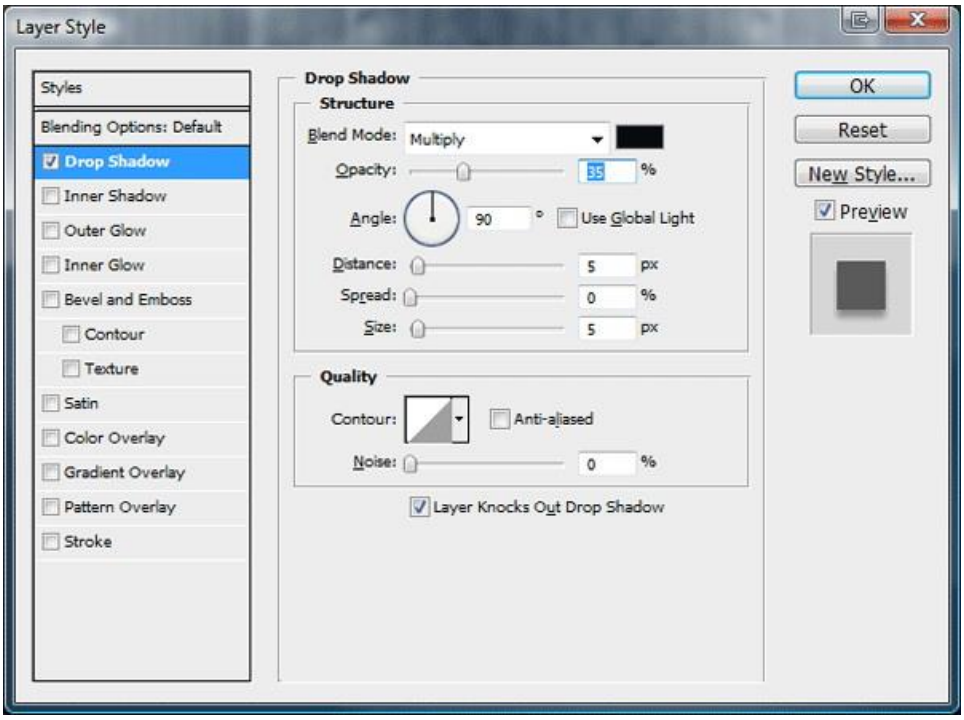
5. Объедините все слои с текстурой и перейдите в меню Тон/Насыщенность (Hue/Saturation), установите значение Тонировать (Colorize) и настройте, как показано ниже.



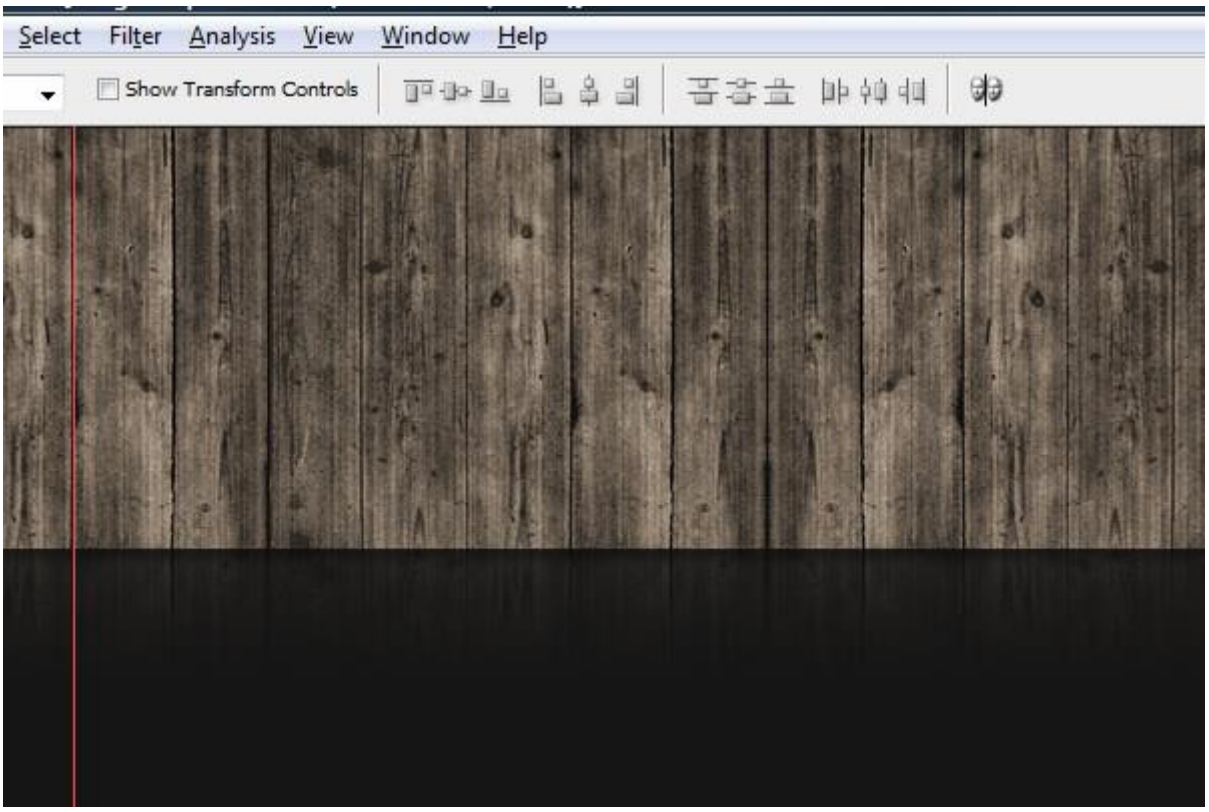
6. Инструментом Прямоугольная область (Rectangular Marquee Tool) создайте выделение деревянной текстуры, оставив нижнюю полоску толщиной в 60 пикселей нетронутой.

7. Вырежьте (Ctrl+X) и вставьте выделение (Ctrl+V) в новом слое. Соедините части текстуры, чтобы она снова стала цельной. Скройте слой с маленькой полоской текстуры и примените стиль Отбрасывание тени (Layer Style - Drop Shadow) к верхнему слою:

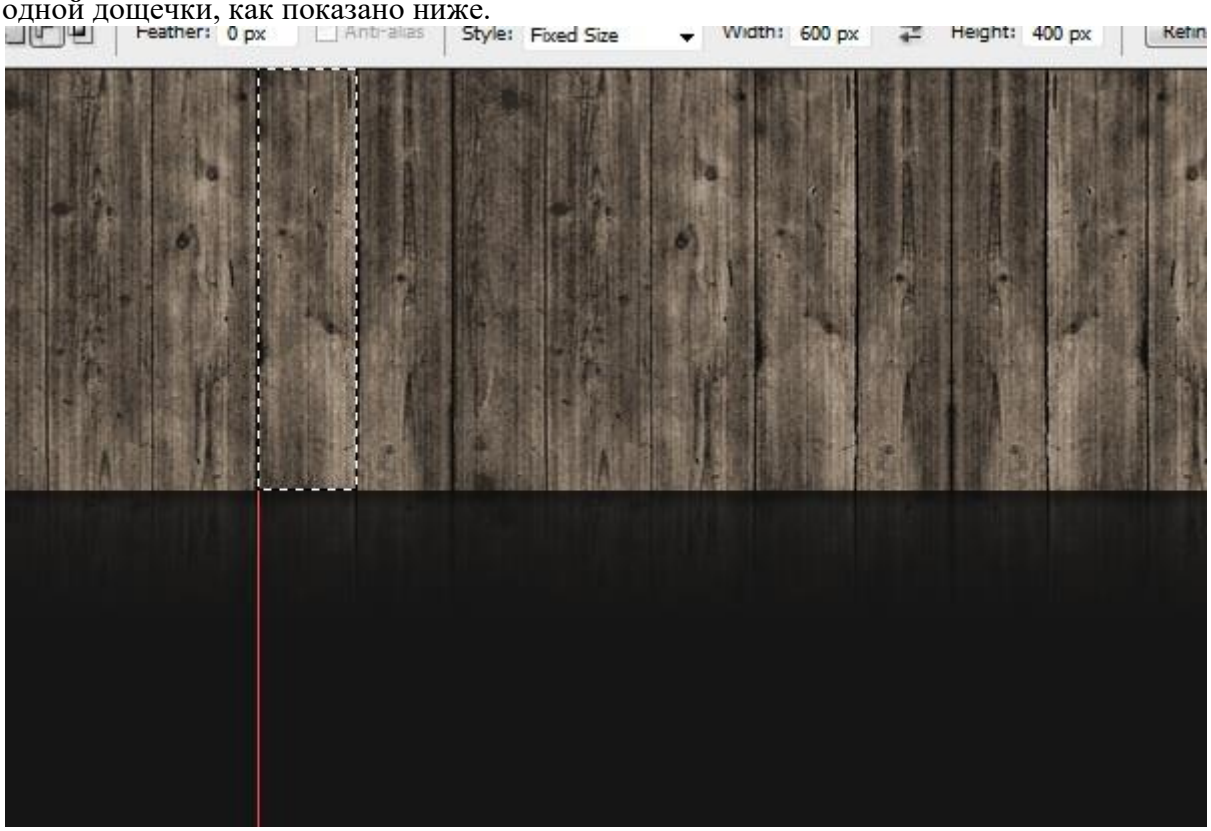
Режим – Умножение



8. Сделайте слой с полоской видимым и добавьте к нему маску слоя (Layer Mask). Сделайте заливку линейным градиентом снизу до половины полоски. Мы получим иллюзию отражения верхней части.

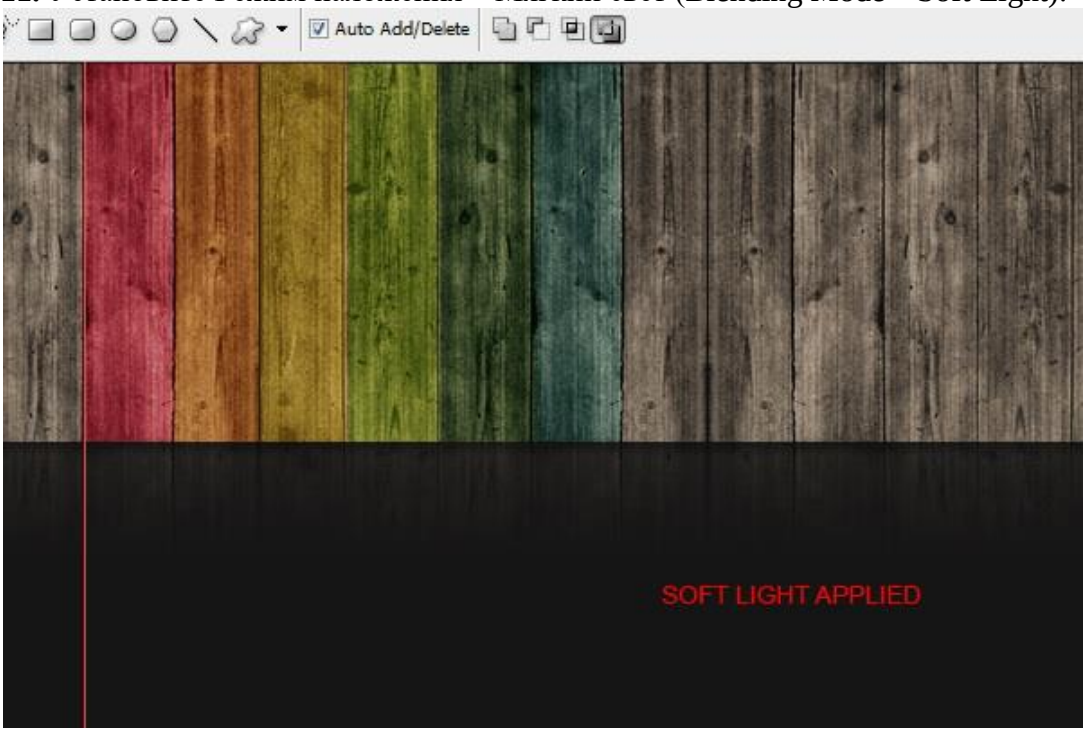


9. Инструментом Прямоугольная область (Rectangular Marquee Tool) создайте выделение одной дощечки, как показано ниже.

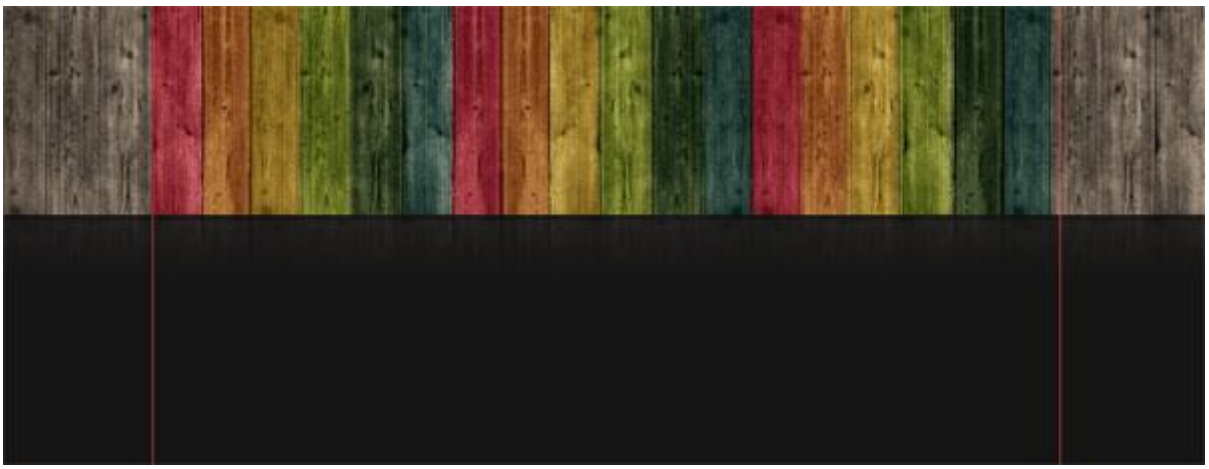


10. Залейте выделение красным цветом, создайте ещё одно выделение и залейте его другим цветом. Повторяйте процесс, пока не получите результат, показанный ниже. Каждую заливку нужно делать в отдельном слое.

11. Установите Режим наложения – Мягкий свет (Blending Mode – Soft Light):



12. Сделайте копию слоёв с цветами и сдвиньте их впаво:



13. В новом слое инструментом Прямоугольная область (Rectangular Marquee Tool) создайте выделение и залейте его чёрным цветом. Напишите в нём первое слово названия сайта. Мой сайт называется “THE DESIGN LAB”. Поэтому первым словом будет “THE”.

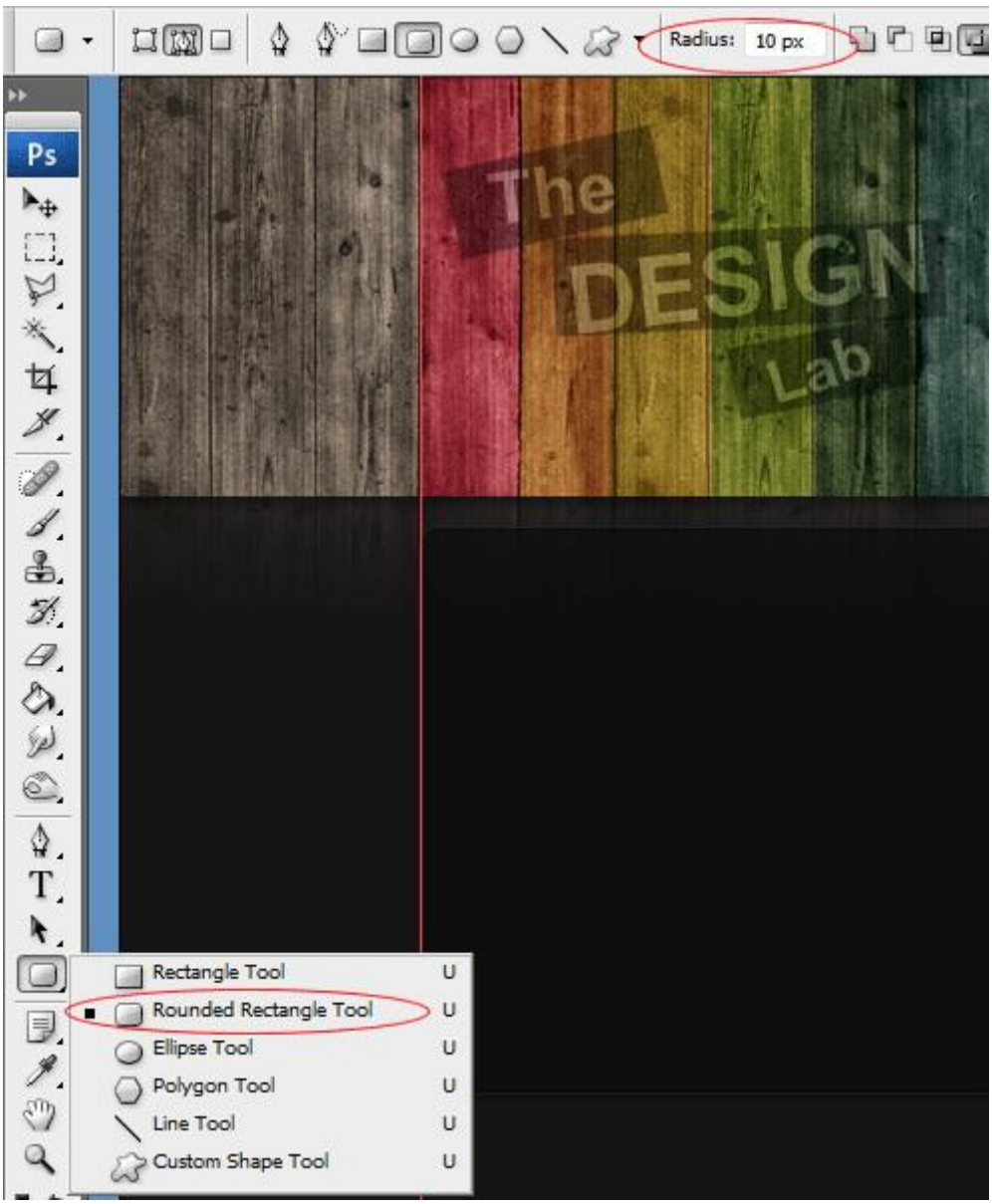
14. Инструментом Свободное трансформирование (Free Transform) наклоните чёрный прямоугольник.

15. Потом наклоните текст, но в противоположную сторону. Установите его Режим наложения – Мягкий свет (Blending Mode – Soft Light). Потом установите Режим наложения слоя с чёрным прямоугольником – Жёсткий свет (Blending Mode – Hard Light) и Непрозрачность – 40%.

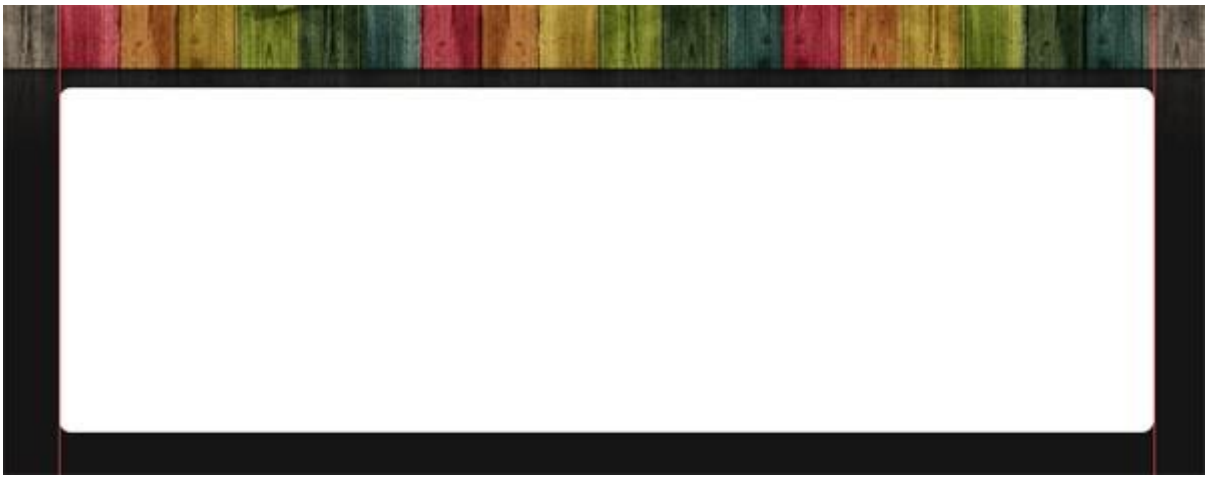
16. Таким образом создайте остальные слова названия сайта. Используйте разные углы наклона.

17. В правом верхнем углу добавьте иконки социальных сетей или любые другие. Я использовал этот набор.

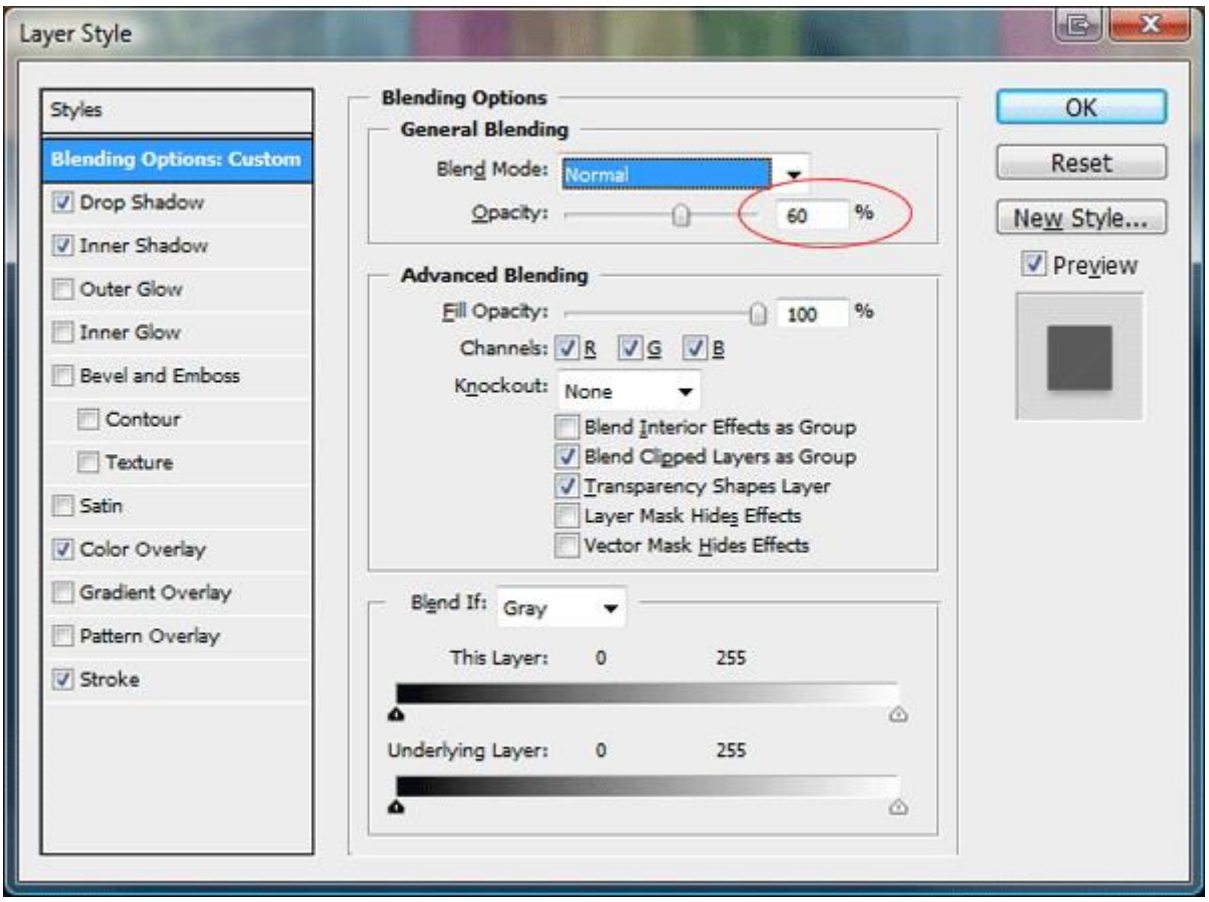
18. Выберите инструмент Прямоугольник с закруглёнными краями (Rounded Rectangle Tool) и настройте, как показано ниже:



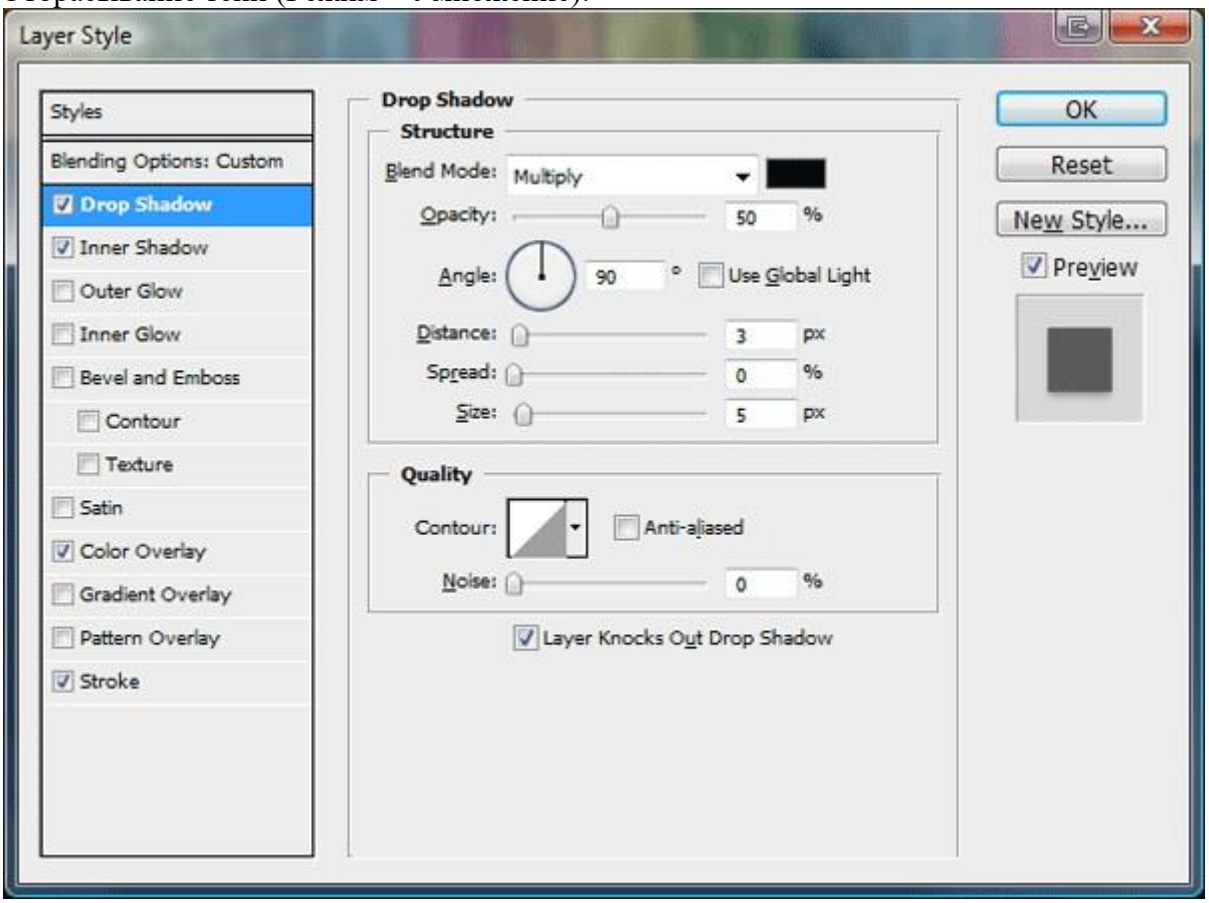
19. Создайте фигуру под областью заголовка сайта. Убедитесь, что его ширина не превышает отведённые пределы.



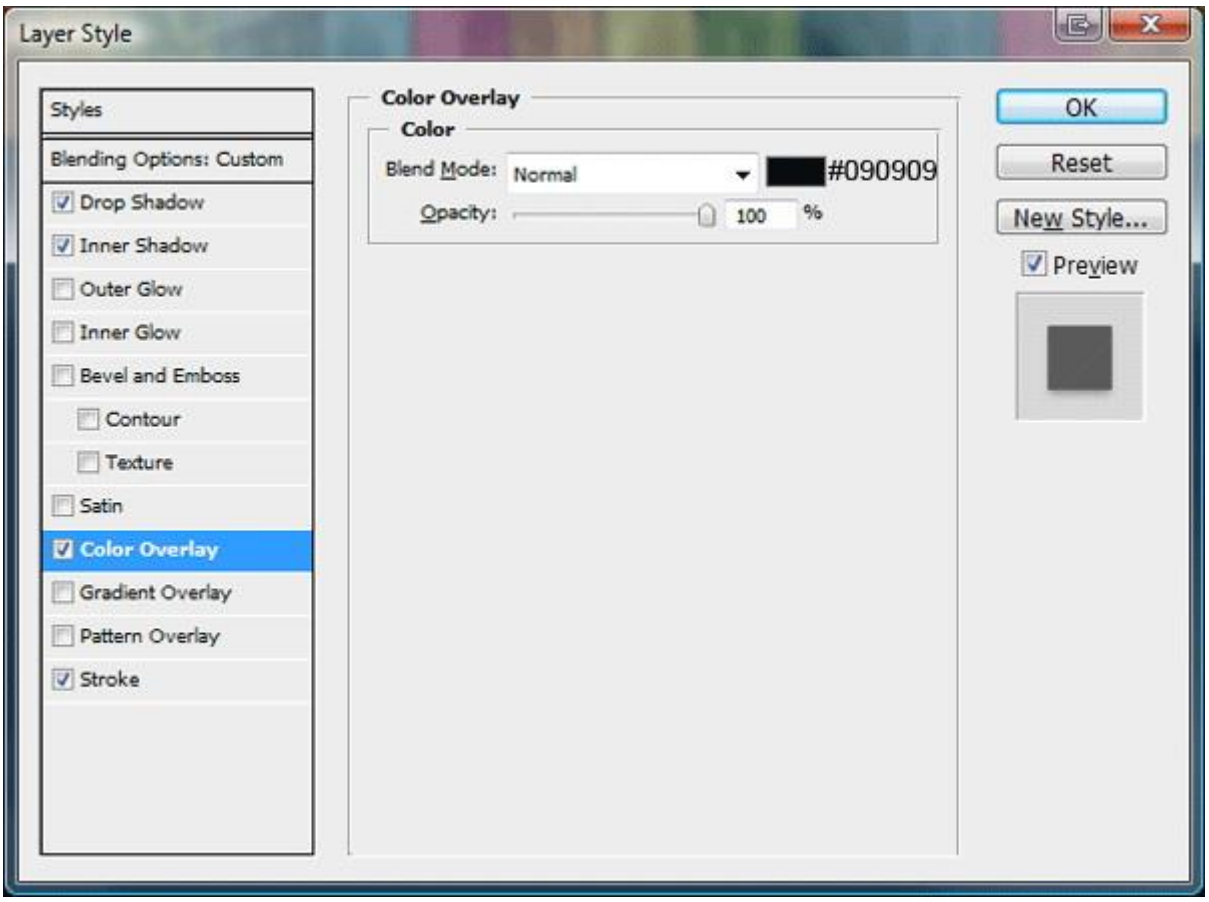
20. Перейдите в меню Параметры наложения (Layer Style - Blending Options) и настройте так:



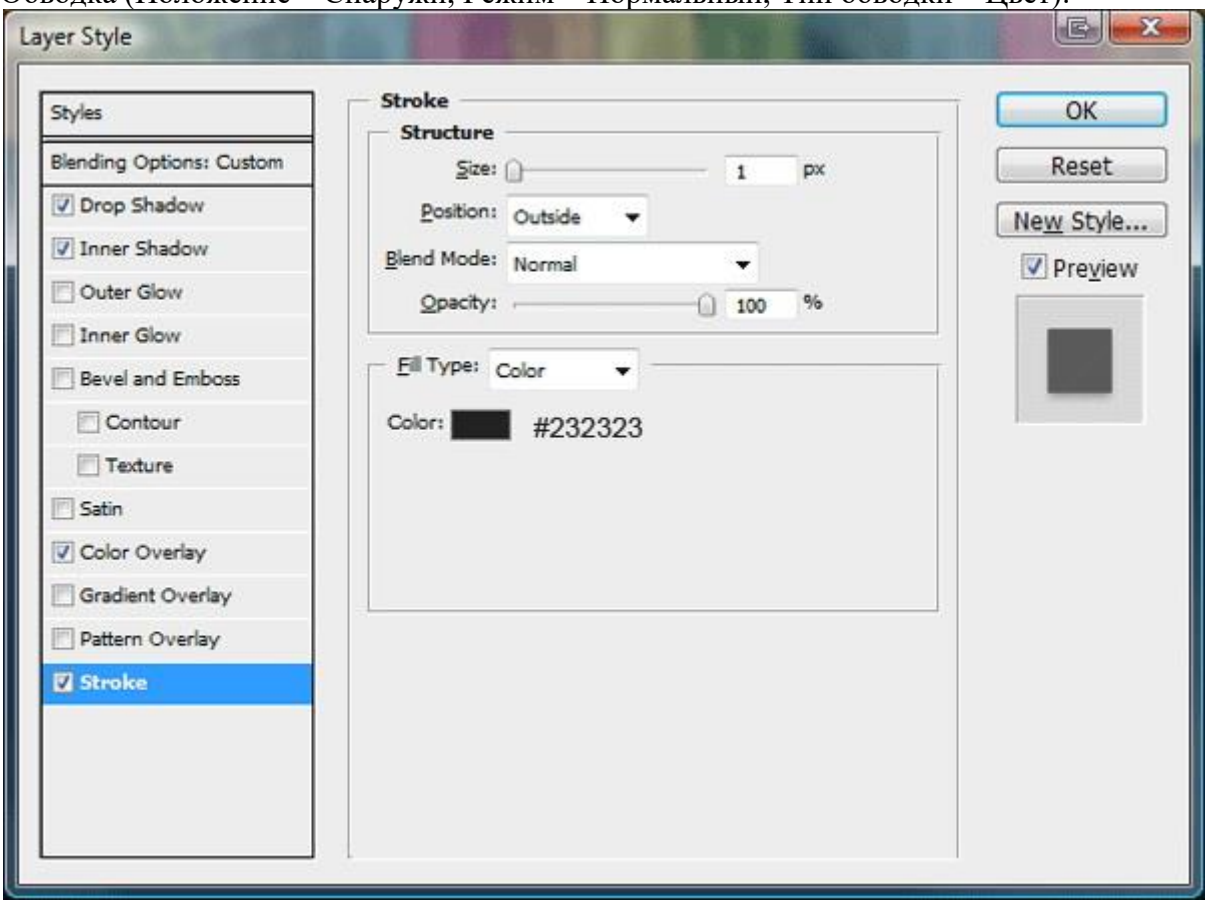
Отбрасывание тени (Режим – Умножение):



Наложение цвета (Режим – Нормальный):



Обводка (Положение – Снаружи, Режим – Нормальный, Тип обводки – Цвет):

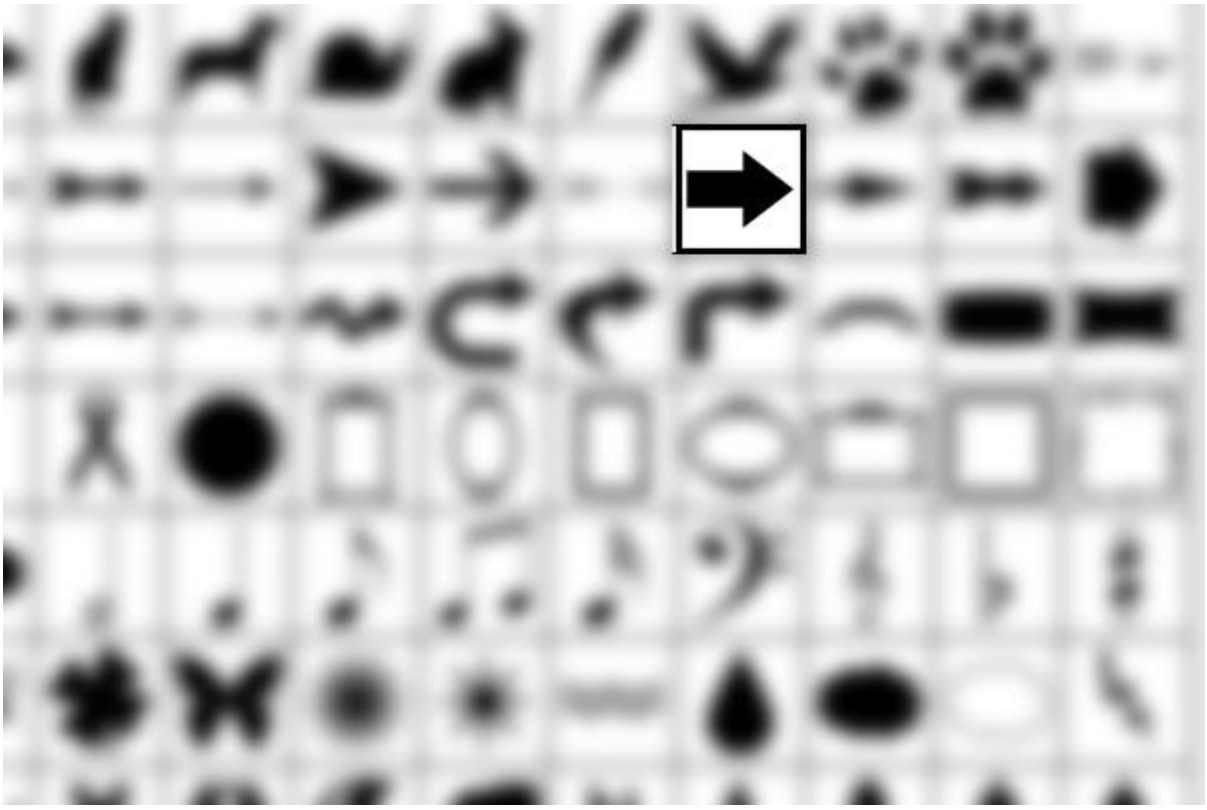


21. Инструментом Прямоугольник (Rectangle Tool) или Прямоугольная область (Rectangular Marquee Tool) создайте чёрный прямоугольник и немного наклоните его в режиме Свободного трансформирования (Free Transform).

22. Сделайте выделение этого слоя с прямоугольником (Ctrl+Click) и примените Сжатие (Modify – Contract) со значением 10 пикселей. Вставьте в это выделение какое-нибудь изображение.

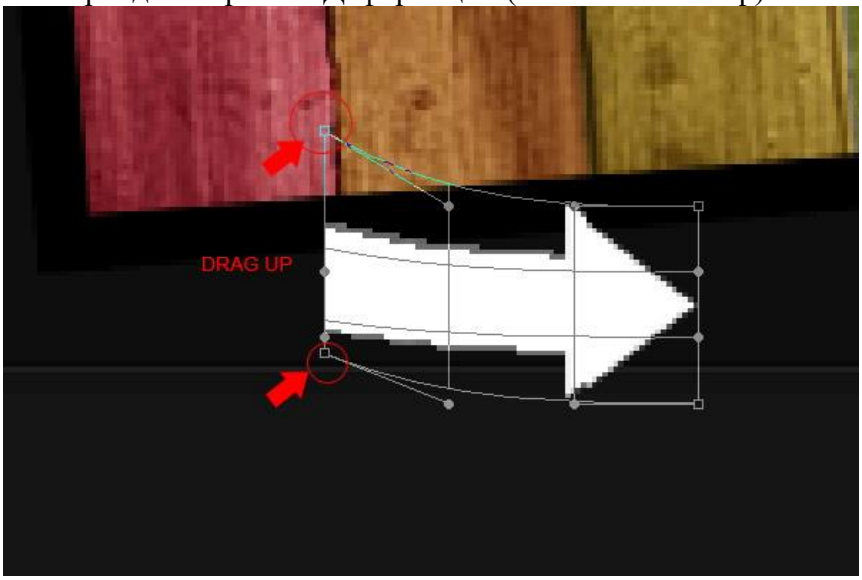
23. Создайте на вставленном изображении свечение, используя инструмент Перо (Pen Tool).

Теперь можно переходить к созданию стрелок. Выберите инструмент Произвольная фигура (Custom Shape Tool) и выберите стрелку, показанную ниже.



24. Создайте фигуру белого цвета где-нибудь на холсте и создайте выделение, как показано ниже:

25. Перейдите в режим Деформации (Transform – Warp) и потяните левые точки вверх.



26. Установите Непрозрачность слоя – 60% и примените стиль Отбрасывание тени (Layer Style - Drop Shadow) со стандартными настройками. Сделайте копию стрелки и поместите её на другую сторону. Справа напишите какой-нибудь текст.

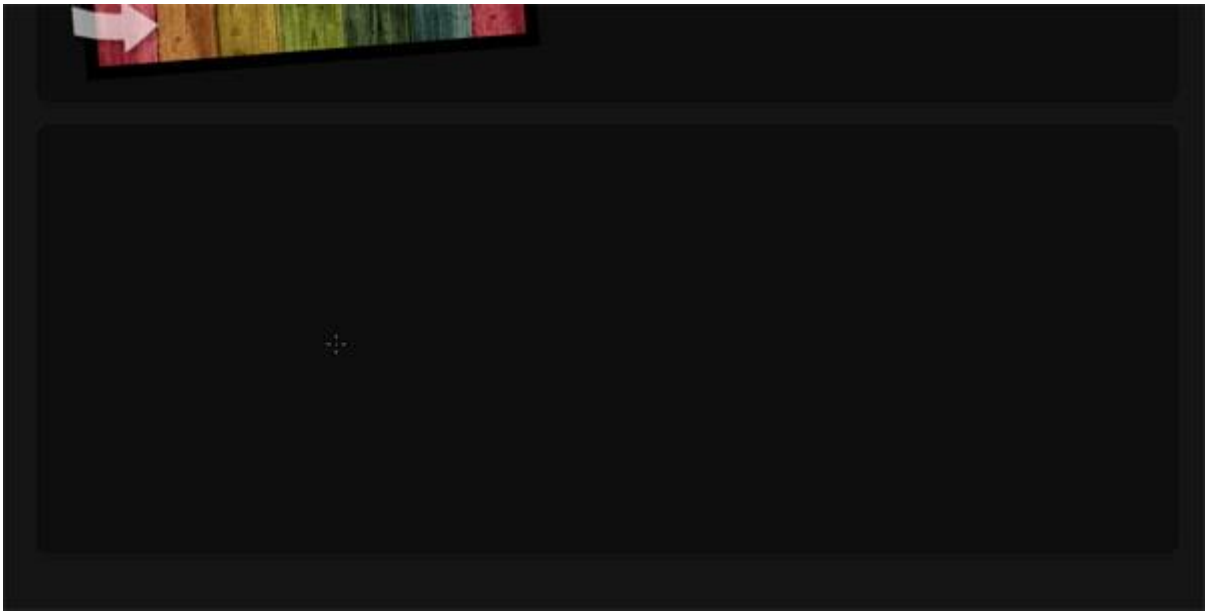
27. Перейдём к созданию навигационного меню. Инструментом Прямоугольник с закруглёнными краями (Rounded Rectangle Tool) с радиусом 10 пикселей создайте фигуру под предыдущей большой областью под заголовком.

28. Сделайте копию слоя несколько раз:

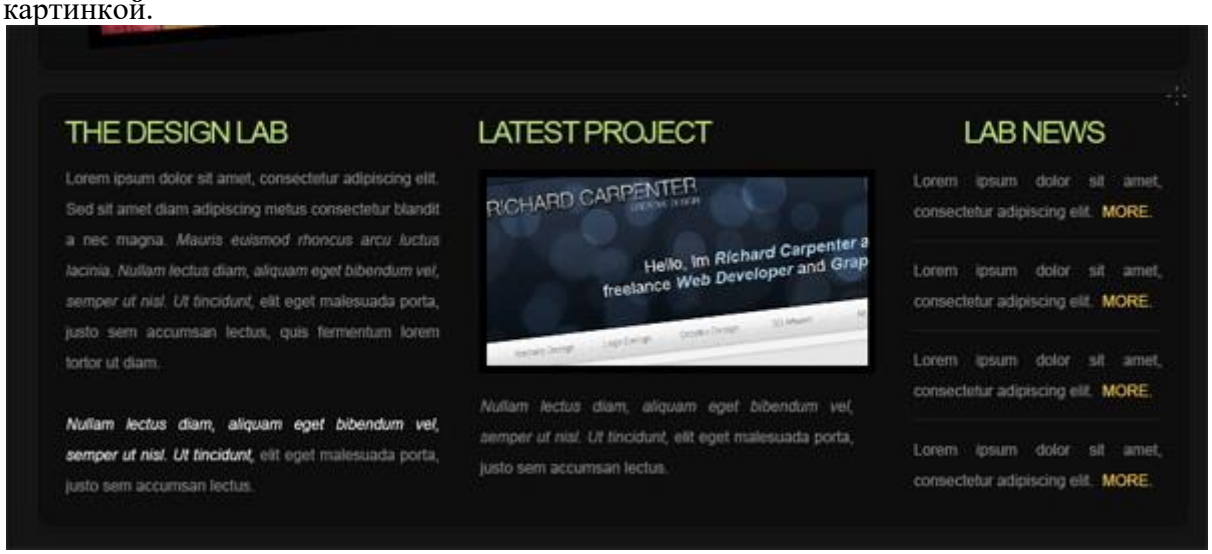
29. Напишите текст на каждой фигуре:

30. Установите Непрозрачность слоёв с кнопками – 70% и измените угол наклона каждой из них. Сделайте выделение слоя с большой областью, на которой мы расположили картинку, стрелки и текст. Потом по очереди выбирайте слои с чёрными прямоугольниками и нажимайте Delete. Это удалит видимую область кнопок позади большого прямоугольника.

31. Создайте большой прямоугольник с закруглёнными краями с радиусом 10 пикселей под предыдущей областью с картинкой. Скопируйте на него все стили слоя с этой области.



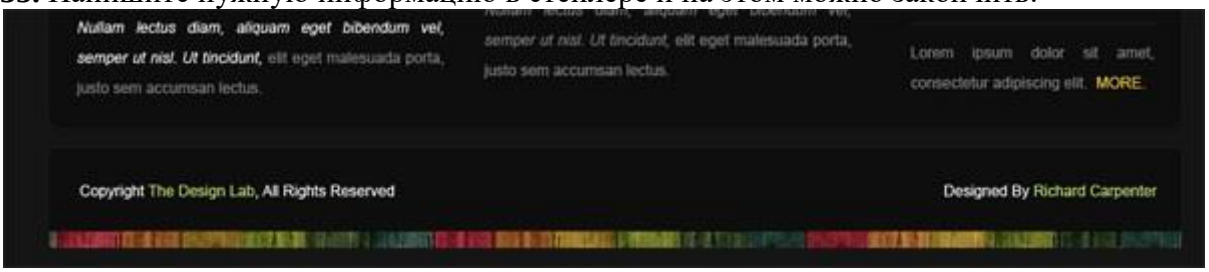
32. Заполните большой прямоугольник контентом. Я создал три колонки с заголовками и картинкой.



33. Создайте обычный прямоугольник в нижней части холста под областью контента. Скопируйте на него все стили с области контента

34. Сделайте выделение цветного заголовка той же ширины, что и создаваемый стейлер (нижняя часть сайта). Высота выделеия – 20-40 пикселей. Потом перейдите в меню Редактировать»Скопировать совмещенные данные (Edit > Copy Merged). Потом вставьте выделение в нижний прямоугольник.

35. Напишите нужную информацию в стейлере и на этом можно закончить:



Финальный результат:

Ссылка на источник урока
http://hv-designs.co.uk/2009/11/11/the-design-lab-2/

Самостоятельная работа

1. Подбор графического материала Web-страницы..
2. Подбор материала для персональной Web-страницы.