Основы web-технологий. Технологии создания web-сайтов.

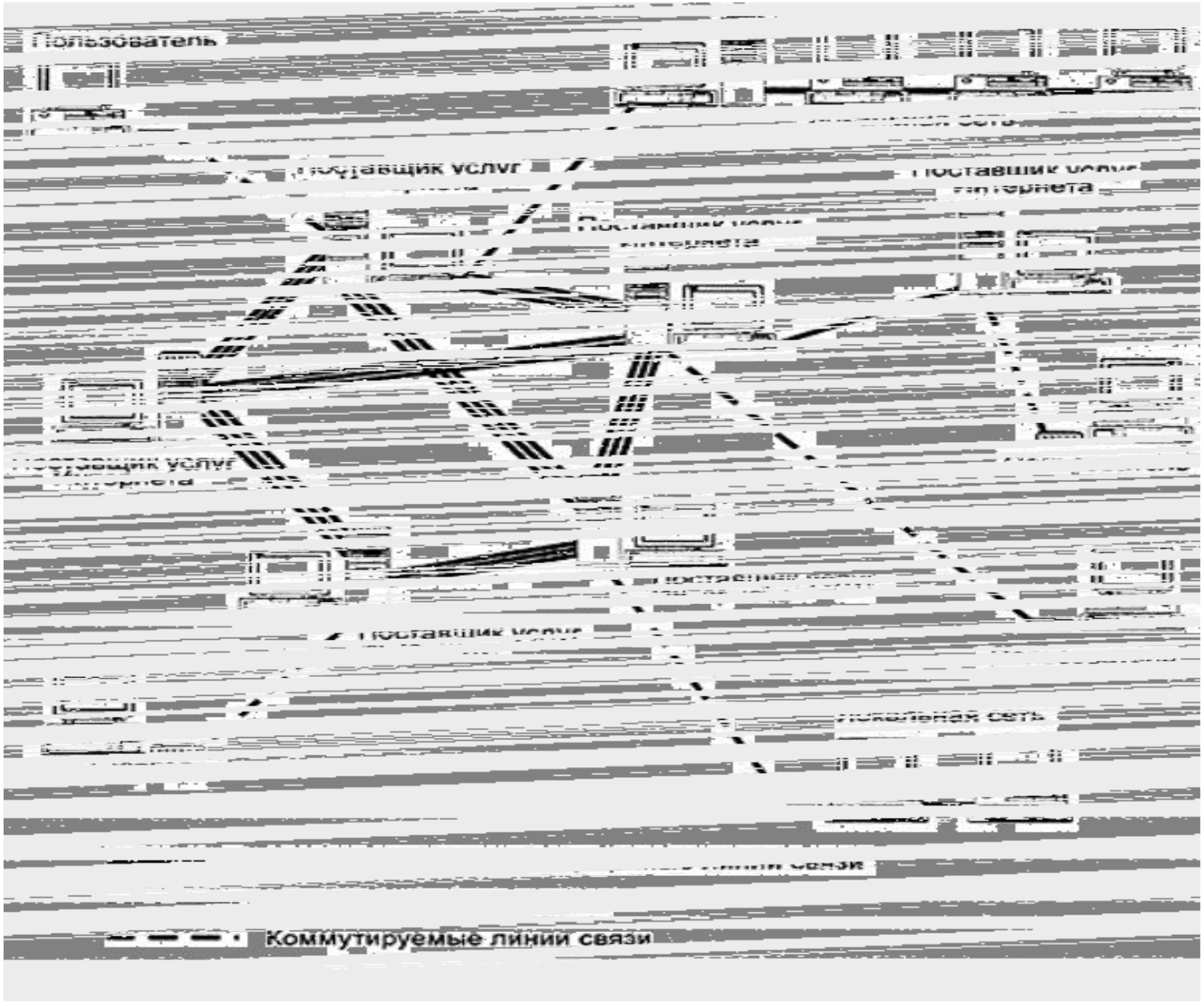
Асирян Александр Вячеславович
ст. преподаватель кафедры АСУ
E-mail: asiryan.a@mail.ru

Сеть Интернет

Интернет — всемирная система объединённых компьютерных сетей, обеспечивающих работу с большим спектром ресурсов (сайты, электронная почта и т.д.).

Сайты — это набор текстовых файлов (гипертекстовых страниц), связанных между собой узлами перехода (гиперссылками для быстрого перехода на другие страницы).

Гипертекст - это текст, в котором содержаться ссылки на другие документы.


Пользователь
Коммутируемые линии связи

Браузер

Веб-обозрева́тель, бра́узер (от англ. Web browser) — программное обеспечение для просмотра веб-сайтов, то есть для запроса веб-страниц (преимущественно из Сети), их обработки, вывода и перехода от одной страницы к другой.

Веб-страница

Веб-страница — документ или информационный ресурс Всемирной паутины, доступ к которому осуществляется с помощью веб-браузера.

Веб-страницы обычно создаются на языках разметки **HTML** и могут содержать гиперссылки для быстрого перехода на другие страницы.

Информация на веб-странице может быть представлена в различных формах:

- текст
- статические и анимированные графические изображения
- аудио
- видео

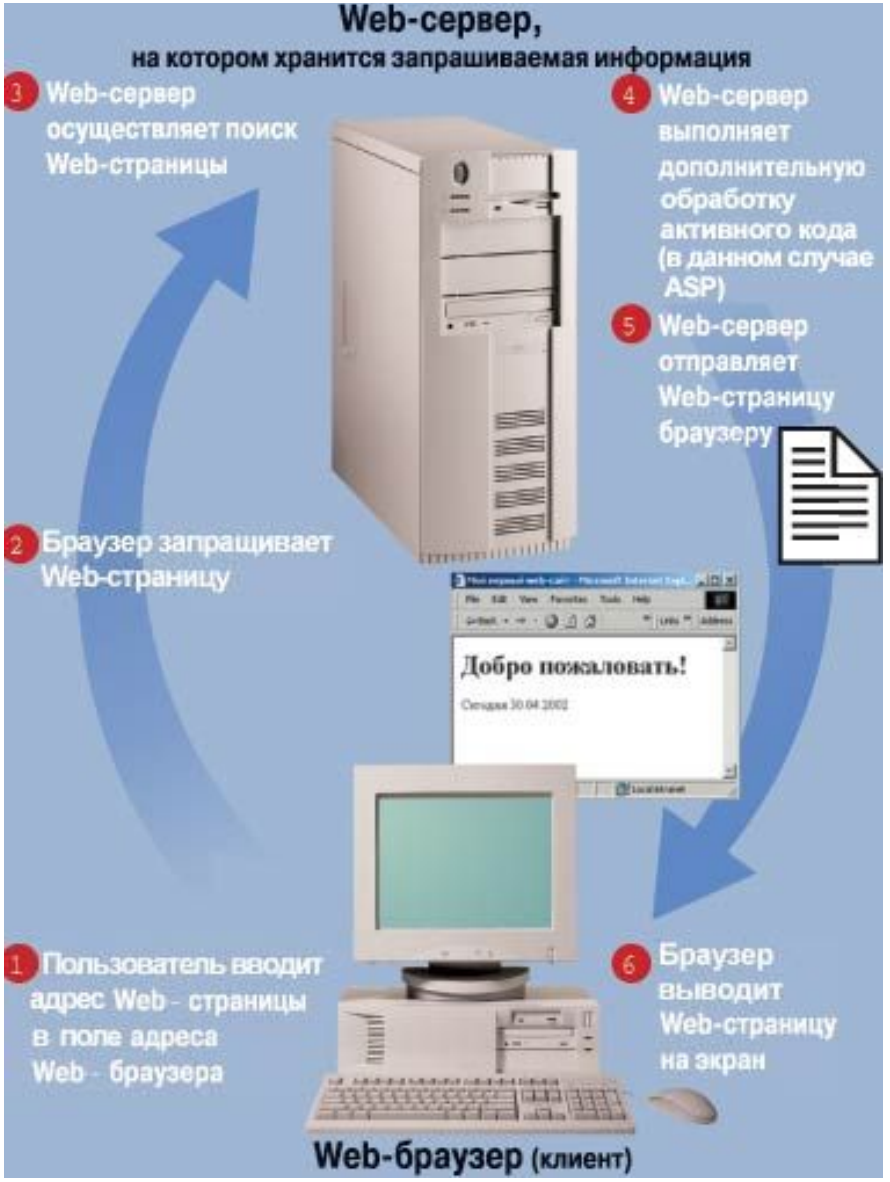
Информационно значимое содержимое веб-страницы обычно называется **контентом.**

Несколько веб-страниц, объединенных общей темой и дизайном, а также связанных между собой ссылками и обычно находящихся на одном сервере, образуют **веб-сайт**.

Клиент/сервер. HTTP-протокол

HTTP (англ. HyperText **Transfer Protocol** — «протокол передачи гипертекста») — протокол прикладного уровня передачи данных (изначально — в виде гипертекстовых документов).

Основой HTTP является технология «клиент-сервер», то есть предполагается существование **потребителей (клиентов),** которые инициируют соединение и посылают запрос, и **поставщиков (серверов),** которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом.



На что тратит время HTTP запрос

Запрос происходит в несколько этапов:

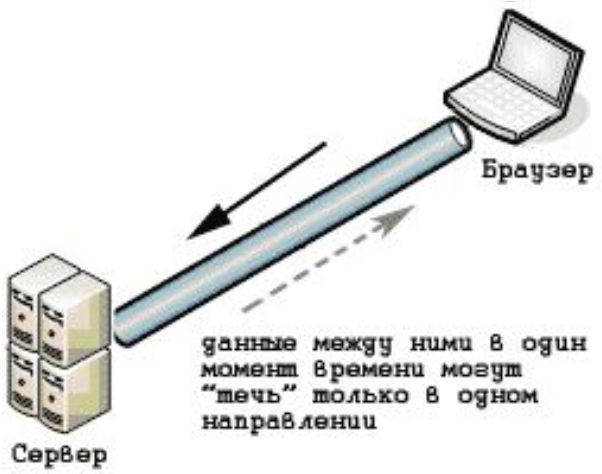
- **DNS-запрос** — поиск ближайшего DNS-сервера, чтобы преобразовать URI (например, google.com) в его числовое представление — IP-адрес (74.125.87.99, прим. — получено посредством команды ping). Это адрес и будет реальным адресом сайта в Интернет.
- **соединение** — установка соединения с сервером по полученному IP-адресу;
- **отправка данных**;
- **ожидание ответа** — ждем пока пакеты данных дойдут до сервера, он их обработает и ответ вернется назад;
- **получение данных**.

Это легко проследить если воспользоваться например, плагином для Firefox — Firebug или встроенными средствами для разработчика в Chrome.

```
+2.27s  Запущен

0                16ms  DNS-запрос
+16ms           375ms  Подключение
+391ms            0    Отправка
+391ms          1.03s  Ожидание
+1.42s            0    Получение

+2.17s  'Содержимое DOM загружено'  (событие)
+5.43s  'загрузка'  (событие)
```



Какие бывают Web-страницы?

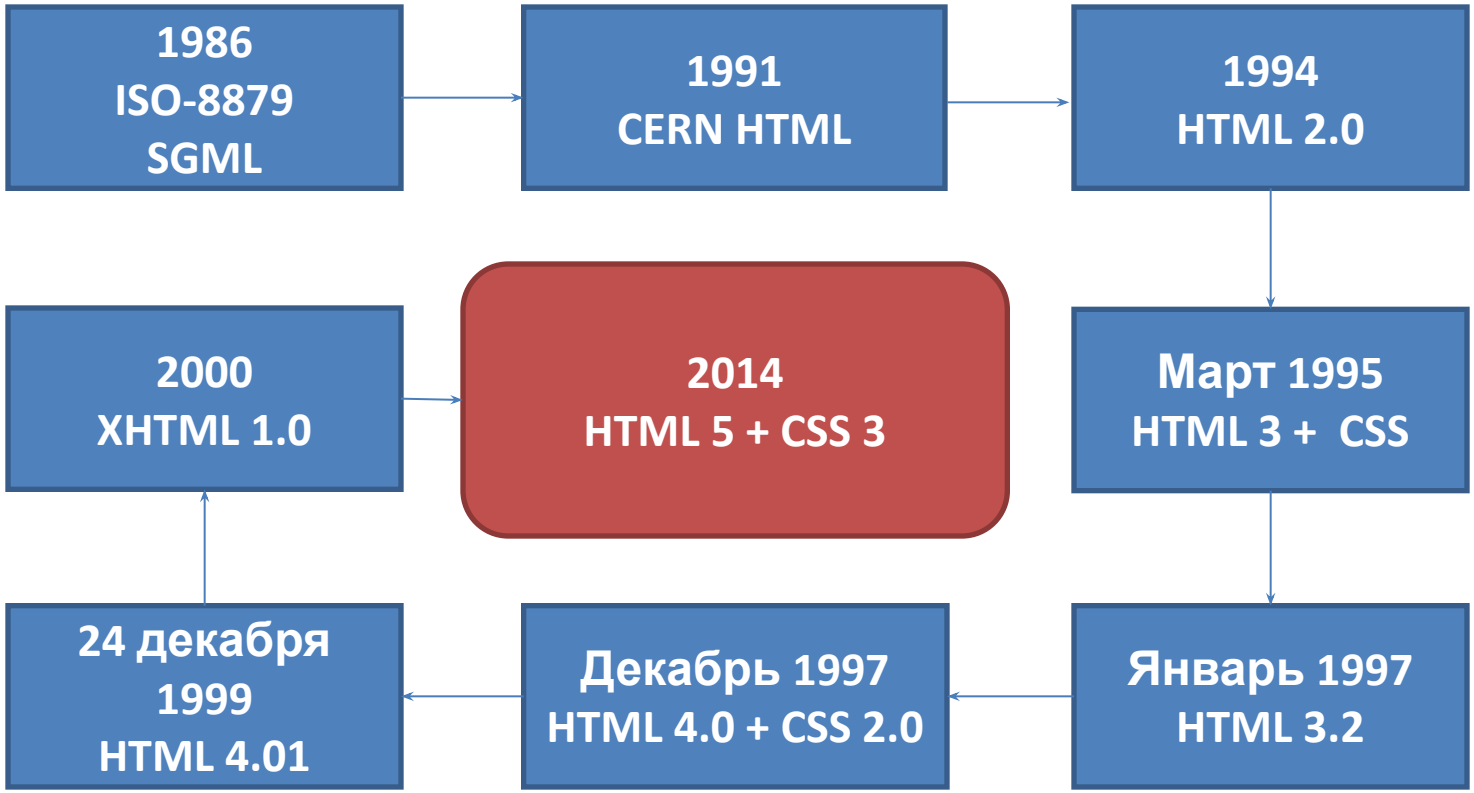
- **статические** – существуют на сервере в виде готовых файлов:
  *.htm, *.html
- **динамические** – полностью или частично создаются на сервере в момент запроса (выбор информации из базы данных)
  *.shtml, *.asp, *.pl, *.php

- позволяют выбирать информацию из базы данных по заранее неизвестным запросам

- дополнительная нагрузка на сервер
- загружаются медленнее

Общее представление о языках гипертекстовой разметки HTML, XML, XHTML

ЭВОЛЮЦИЯ ЯЗЫКОВ РАЗМЕТКИ.



HTML

(от англ. ***HyperText Markup Language*** — «язык разметки гипертекста»)

Основным форматом представления документов в сети Интернет является язык гипертекстовой разметки HTML (стандартный язык разметки документов во Всемирной паутине) .

HTML – это определенная совокупность правил (тегов), по которым оформляется документ. Теги показывают Интернет-браузеру, как следует отображать текст на Web-страничке.

Структура тега (пары тегов) всегда такова:

<название тега> … </название тега>

Теги могут содержать атрибуты, характеризующие отображение информации внутри тега.

XML и XHTML

XML (eXtensible *Markup Language)*

XML похож на язык HTML тем, что для описания различных разделов документа в нем используются тэги. В отличие от HTML, XML позволяет разработчикам определять собственные тэги и ставить в соответствие им способы воспроизведения информации. XML-тэги чувствительны к регистру символов.

XHTML (eXtensible *HyperText Markup Language*)

XHTML - это основанный на XML язык разметки гипертекста, максимально приближенный к стандартам HTML. XHTML отличается от HTML строгостью написания кода. Если HTML позволял писать практически любые конструкции и браузер их корректно распознавал, то теперь, с появлением XHTML, это стало невозможным. Строгие требования к оформлению XHTML-кода позволяют избежать многих ошибок ещё на стадии написания и отладки.

На данный момент используется HTML 5. он был создан как единый язык разметки, который мог бы сочетать синтаксические нормы HTML и XHTML.

Тэги

Тэг – это команда языка HTML, которую выполняет браузер:

- непарные тэги

вставить рисунок

```
<IMG SRC = "vasya.jpg">
```

- парные тэги (*контейнеры*)

открывающий

```
<TABLE>
...
</TABLE>
```

область действия тэга: описание таблицы

закрывающий

Структура Web-страницы

```
<HTML>
  <HEAD>
   <TITLE>
   </TITLE>
  </HEAD>
   <BODY>
   </BODY>
</HTML>
```

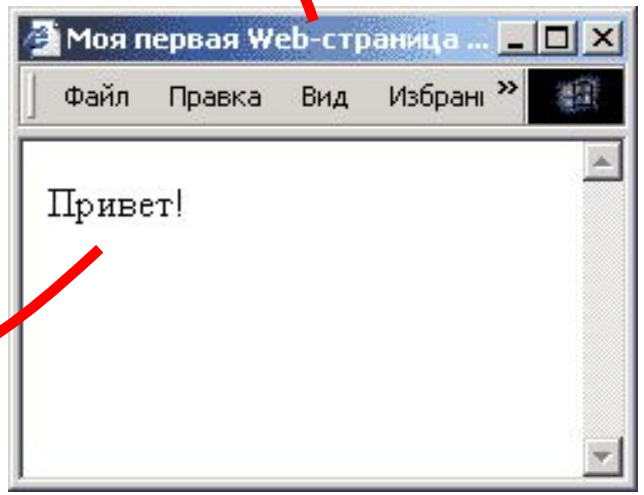
- HTML-код страницы помещается внутрь контейнера **<HTML> …</HTML>**
- Заголовок Web-страницы заключается в контейнер <HEAD>…</HEAD>
- Основное содержание страницы помещается в контейнер <BODY>…</BODY>
- Название Web-страницы содержится в контейнере <TITLE>…</TITLE> и выводится в строке заголовка браузера.

Простейшая Web-страница

first.html

```
<HTML>
<HEAD>
<TITLE>Моя первая
   Web-страница</TITLE>
</HEAD>
<BODY>
Привет!
</BODY>
</HTML>
```

шапка («голова»)



основная часть («тело»)

Примеры уроков

http://sevidi.narod.ru/

http://www.trend-life.ru/obuch/html-1.php

Создание файла веб-страницы

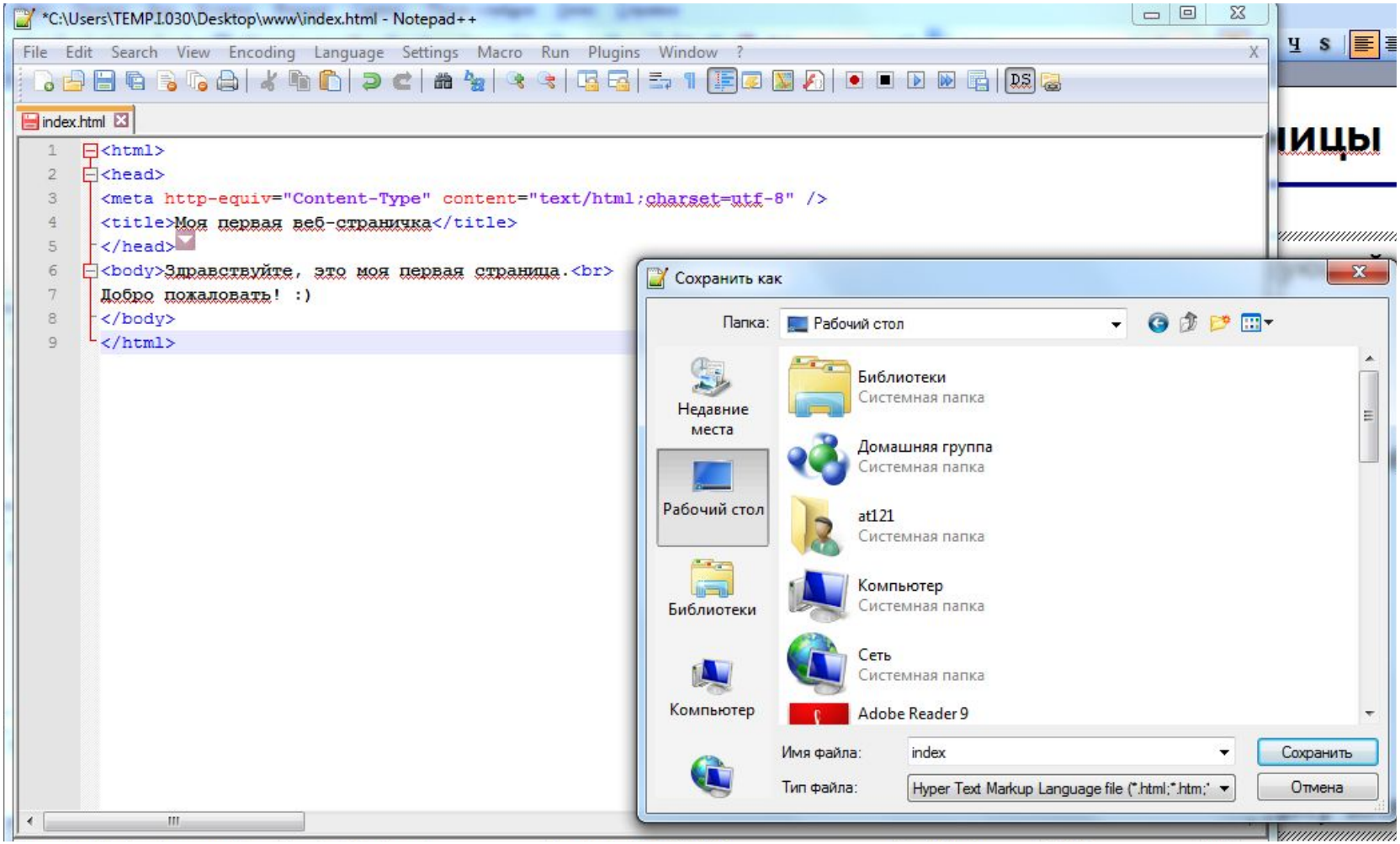
Открыть **Notepad++** и введите туда следующий текст:

```
<html>
<head>
<title>Моя первая веб-страничка</title>
<meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
</head>
<body>
Здравствуйте, это моя первая страница.<br>
Добро пожаловать! :)
</body>
</html>
```

Сохраним этот документ, присвоив ему имя *.html

Сохранение

Выбрать в меню Файл - Сохранить как…, и сохранить файл с расширение .html

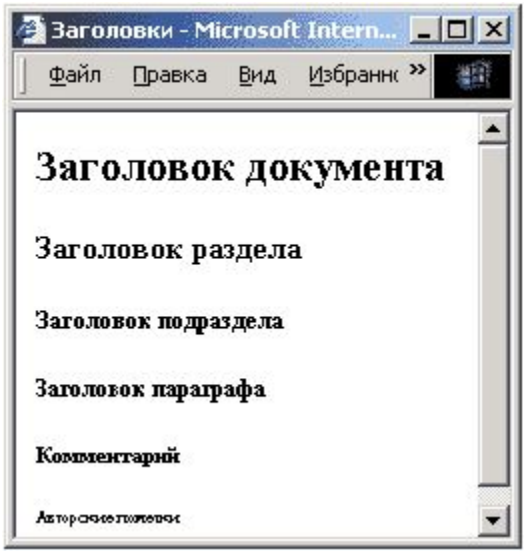


Web-страницы. Язык HTML

Тема 2. Оформление текста

Заголовки: H1 … H6

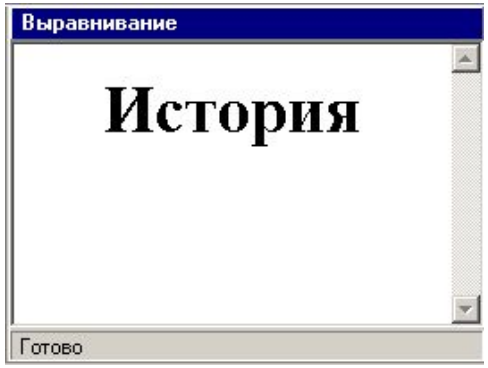
```
<BODY>
<H1>Заголовок документа</H1>
<H2>Заголовок раздела</H2>
<H3>Заголовок подраздела</H3>
<H4>Заголовок параграфа</H4>
<H5>Комментарий</H5>
<H6>Авторские пометки</H6>
</BODY>
```



выравнивание:

left,
center,
right

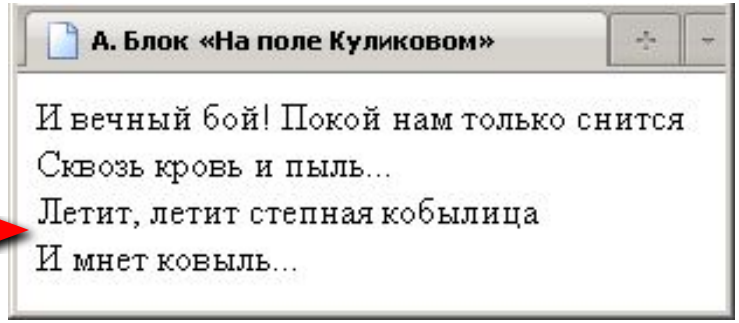
```
<H1 ALIGN=center>История</H1>
```



Абзацы

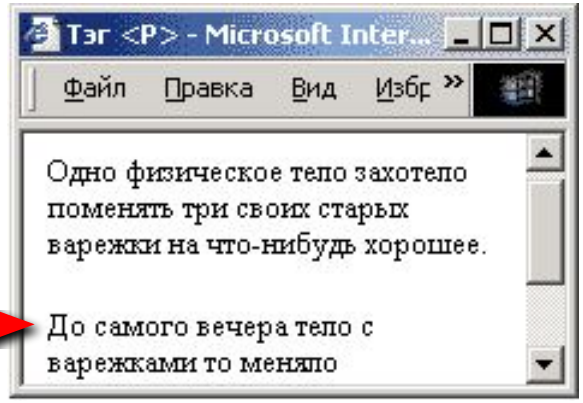
- переход на новую строку

```
И вечный бой! Покой
нам только снится<BR>
Сквозь кровь и пыль...<BR>
Летит, летит степная
кобылица<BR>
И мнет ковыль...
```



- абзац (с отступами)

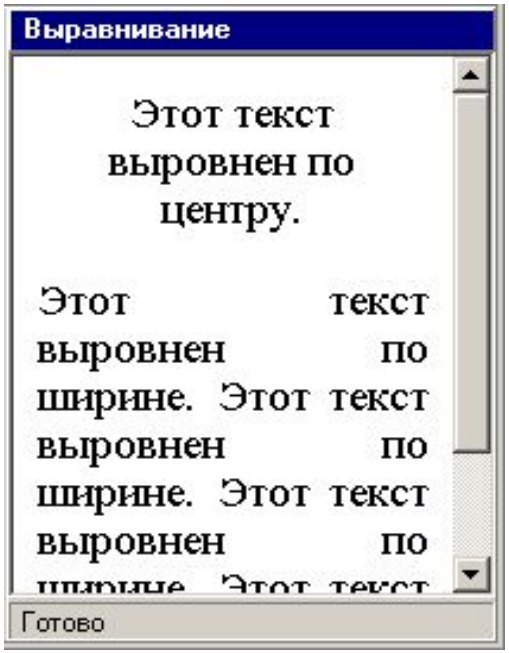
```
<P>
Одно физическое тело захотело
поменять три своих старых варежки
на что-нибудь хорошее.
</P>
<P>
До самого вечера тело с
варежками ...
</P>
```



Выравнивание

атрибут тэга <P>

```
<P ALIGN="center">
Этот текст выровнен по центру.
</P>
<P ALIGN="justify">
Этот текст выровнен по ширине.
Этот текст выровнен по ширине.
Этот текст выровнен по ширине.
Этот текст выровнен по ширине.
</P>
```



left по левой границе
right по правой границе
center по центру
justify по ширине

Физическая разметка

курсив (*italic*)	`<I>Вася</I>`	*Вася*
жирный (*bold*)	`<B>Вася</B>`	**Вася**
подчеркивание (*underline*)	`<U>Вася</U>`	<u>Вася</u>
зачеркивание (*strike out*)	`<S>Вася</S>`	~~Вася~~
верхний индекс (*superscript*)	`Вася<SUP>2</SUP>`	Вася^{2}
нижний индекс (*subscript*)	`Вася<SUB>2</SUB>`	Вася$_{2}$

Специальные символы

Символ	*HTML-код*	*Название*
–	&mdash;	(длинное) тире
		неразрывный пробел
§	&sect;	параграф
©	&copy;	символ авторского права
«	&laquo;	левая русская кавычка
»	&raquo;	правая русская кавычка
®	&reg;	зарегистрированная торговая марка
°	&deg;	градус
²	&sup2;	квадрат
³	&sup3;	куб
¼	&frac14;	четверть
½	&frac12;	половина
¾	&frac34;	три четверти
×	&times;	знак умножения
÷	&divide;	знак деления

Web-страницы. Язык HTML

Тема 3. Оформление документа.

Тэг BODY – общие свойства страницы

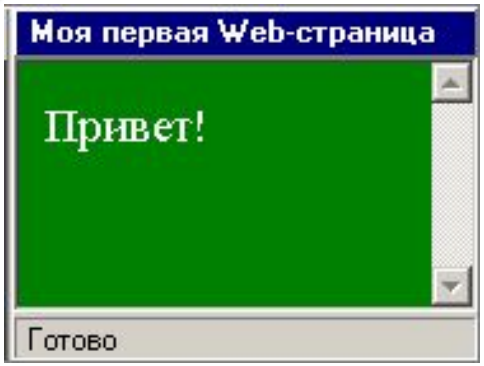
- **цвет фона и текста**

атрибуты тэга

цвет текста

```
<BODY TEXT="white"
      BGCOLOR=#00FF00>
Привет!
</BODY>
```

цвет фона

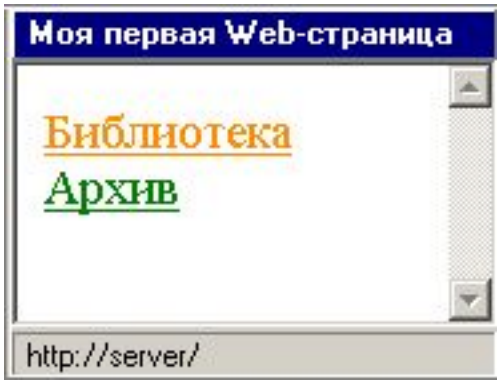


- **цвет гиперссылок**

цвет ссылок

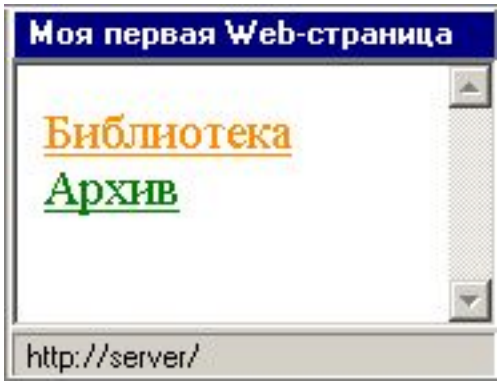
```
<BODY LINK="#FF8C00"
      VLINK=green>
...
</BODY>
```

посещенные ссылки *(visited link)*



Цвет гиперссылок

```
<BODY LINK="#FF8C00"
      VLINK=green
      ALINK=red>
...
</BODY>
```

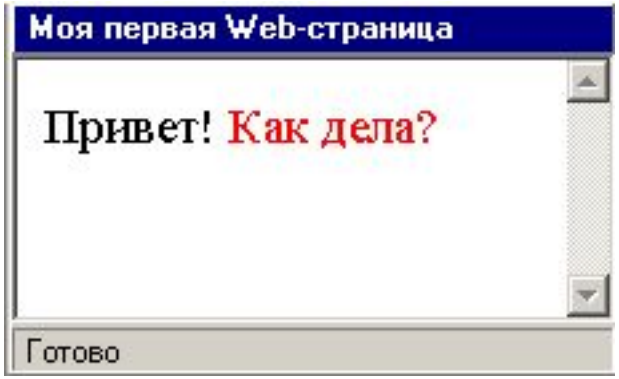


LINK	ссылки, на которых не были
VLINK	посещенные ссылки
ALINK	активные ссылки

Тэг FONT – свойства блока текста

- цвет текста

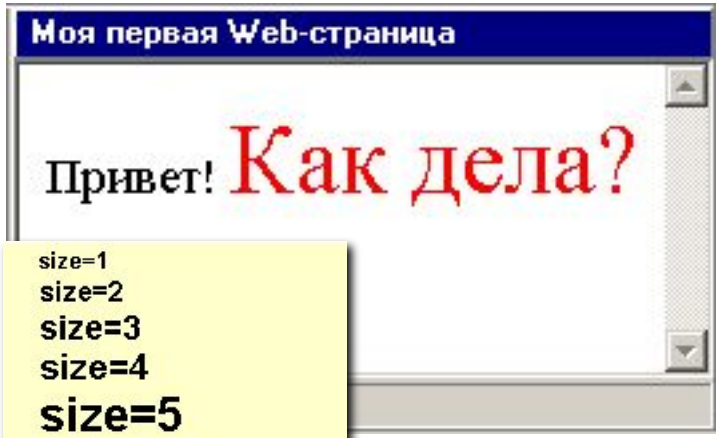
```
Привет!
<FONT COLOR=red>
Как дела?
</FONT>
```



- размер шрифта

```
Привет!
<FONT COLOR=red
      SIZE=6>
Как дела?
</FONT>
```

от 1 до 7
(3 – нормальный)

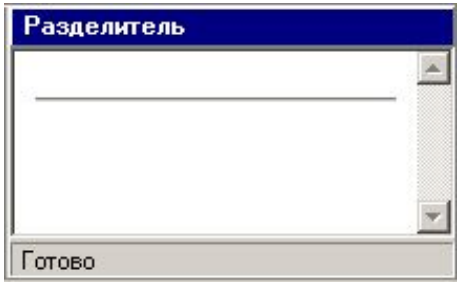


size=1
size=2
size=3
size=4
size=5
size=6
size=7

Линия-разделитель

horizontal rule

```
<HR>
```

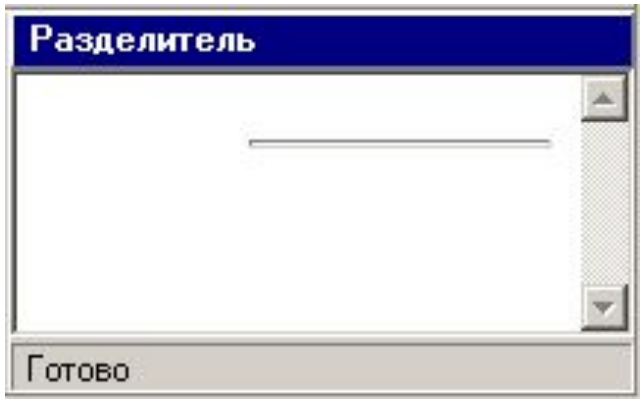


ширина в пикселях или процентах

толщина

выравнивание

```
<HR WIDTH="60%" SIZE="3" ALIGN="right">
```



Не рекомендуется использовать – лучше заголовки разделов!

Кодирование цвета

- имена

  red, *green*, *blue*, *magenta*, *black*, *white*

- шестнадцатеричные коды

#FA8072

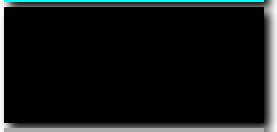
R G B

#FF0000

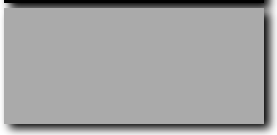
#FFFFFF

#00FFFF

#000000



#AAAAAA



Web-страницы. Язык HTML

Тема 4. Рисунки

Форматы рисунков

GIF *(Graphic Interchange Format)*

- сжатие без потерь
- прозрачные области
- анимация
- только с палитрой (2…256 цветов)

рисунки с четкими границами, мелкие рисунки

JPEG *(Joint Photographer Expert Group)*

- сжатие с потерями
- только *True Color* (16,7 млн. цветов)
- нет анимации и прозрачности

рисунки с размытыми границами, фото

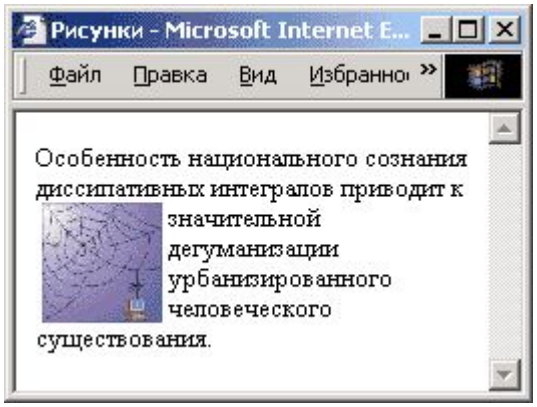
PNG *(Portable Network Graphic)*

- сжатие без потерь
- с палитрой (PNG-8) и *True Color* (PNG-24)
- прозрачность и полупрозрачность (альфа-канал)
- нет анимации
- плохо сжимает мелкие рисунки

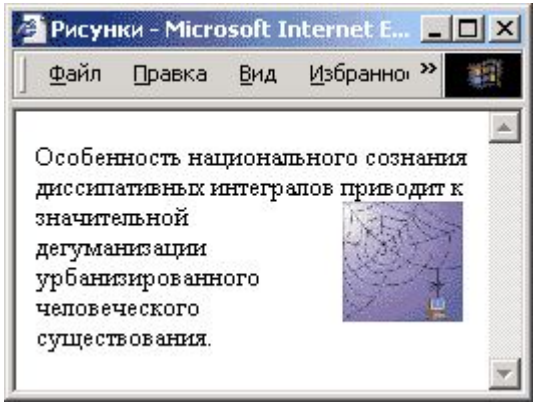
Выравнивание

```
<IMG SRC="flag.jpg" ALIGN="left">
```

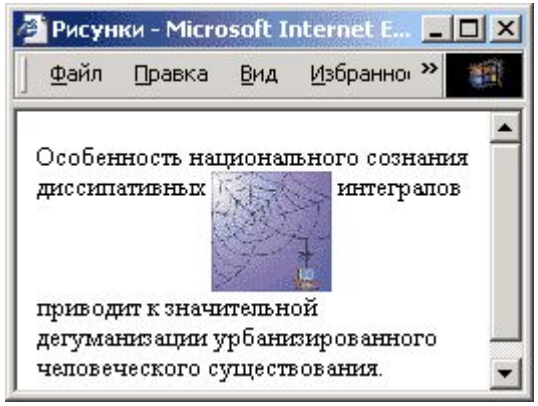
left



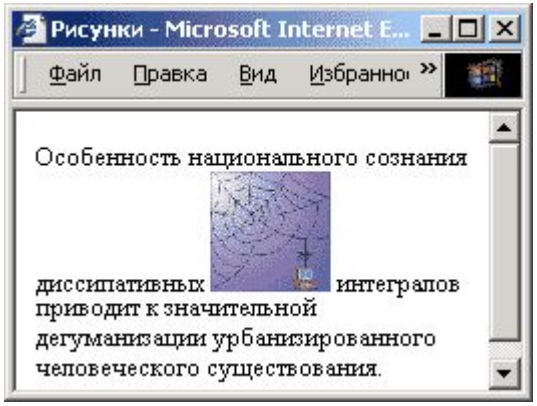
right



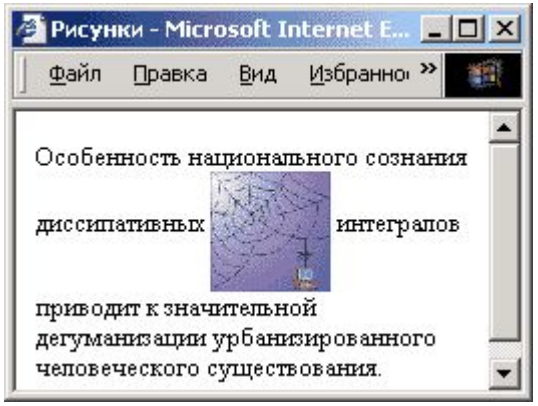
top



bottom
(по умолчанию)



middle



picture - Блокнот

Файл Правка Формат Справка

```
<html>
<title>Пример с рисунком</title>
<body>
<img src='picture.jpg' align='left'>
<p>По умолчанию после вставки изображения
следующая строка оказывается после рисунка.
Если есть необходимость, чтобы текст обтекал
рисунок, можно воспользоваться параметром align
для тега img.
</body>
</html>
```

Пример с рисунком - Microsoft Internet Explorer

Файл Правка Вид Избранное Сервис Справка

Назад Поиск Избранное

Адрес C:\Documents and Settings\Администратор\Рабочий сто

По умолчанию после вставки изображения следующая строка оказывается после рисунка. Если есть необходимость, чтобы текст обтекал рисунок, можно воспользоваться параметром align для тега img.

Готово Мой компьютер

Рисунки в документе

из той же папки:

image *(изображение)*

source *(источник)*

```
<IMG SRC="flag.jpg">
```

из другой папки:

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

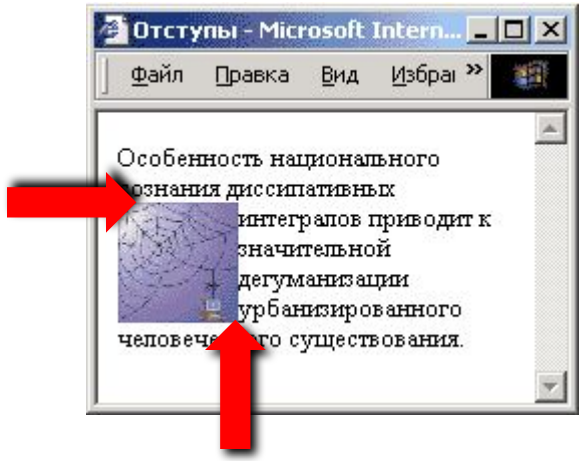
```
<IMG SRC="../images/flag.jpg">
```

с другого сервера:

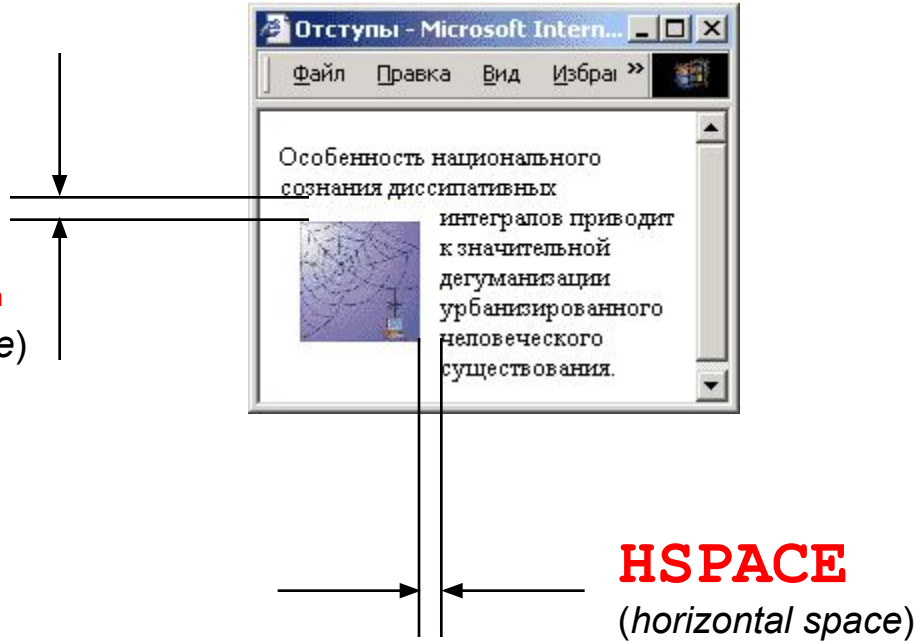
```
<IMG SRC="http://www.vasya.ru/img/flag.jpg">
```

Отступы

```
<IMG SRC="net.jpg"
     ALIGN="left">
```




```
<IMG SRC="net.jpg"
     ALIGN="left"
     HSPACE=10
     VSPACE=10>
```



Другие атрибуты

```
<IMG SRC="myphoto.jpg"
   ALT="Моя фотография"
      WIDTH=100 HEIGHT=150
   BORDER=0>
```

- всплывающая подсказка
- надпись на месте рисунка, если его нет

размеры позволяют:
- растянуть - сжать
- не портить дизайн, если рисунка нет

толщина рамки вокруг рисунка

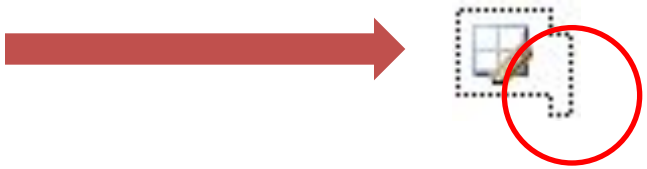
Рисунок-гиперссылка

локальная ссылка:

```
<A HREF="table.htm">
<IMG SRC="tbl.jpg" ALT="Таблицы" BORDER=0>
</A>
```

иначе будет синяя рамка вокруг

если </A> не вплотную к <IMG ...>, будет «хвост»



ссылка на другой сервер:

```
<A HREF="http://www.mail.ru">
<IMG SRC="mailru.jpg"
     ALT="Бесплатная почта" BORDER=0></A>
```

не будет «хвоста»

Web-страницы. Язык HTML

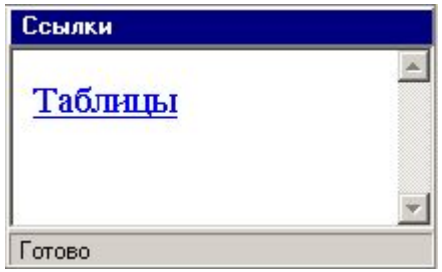
Тема 5. Гиперссылки

Ссылки на другие страницы сайта

- страница в той же папке

anchor (якорь)

```
<A HREF="table.htm">Таблицы</A>
```



hyper reference (гиперссылка)

- страница во вложенной папке

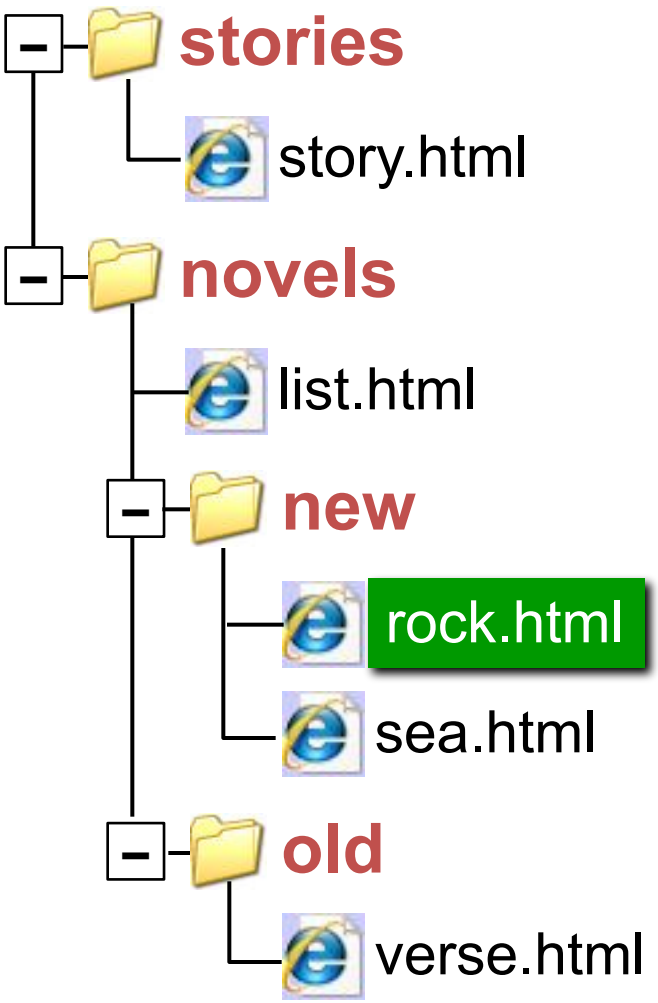
```
<A HREF="example/ex1.htm">Пример</A>
```

- страница в соседней папке

```
<A HREF="../texts/text1.htm">Текст</A>
```

выйти из текущей папки

Примеры (ссылки из файла ***rock.html***)



Ссылки на другие сайты

- на главную страницу сайта

```
<A HREF="http://www.mail.ru">Почта</A>
```

index.htm, index.html, default.asp, …

- на конкретную страницу сайта (URL)

```
<A HREF="http://www.vasya.ru/text/a.htm">
Васин текст</A>
```

- на файл для скачивания

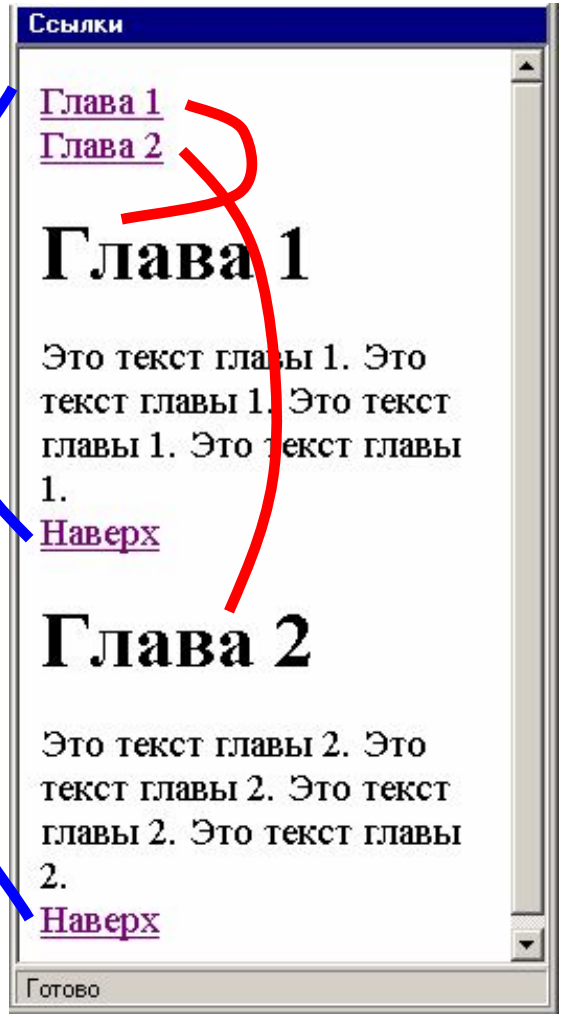
```
<A HREF="http://www.vasya.ru/prog.zip">
Скачать</A>
```

Ссылки внутри страницы

```
<A NAME="up"></A>
<A HREF="#chap1">Глава 1</A><br>
<A HREF="#chap2">Глава 2</A><br>
<A NAME="chap1"></A>
<H1>Глава 1</H1>
Это текст главы 1. Это текст главы 1.
Это текст главы 1. Это текст главы
1.<BR>
<A HREF="#up">Наверх</A>
<A NAME="chap2"></A>
<H1>Глава 2</H1>
Это текст главы 2. Это текст главы 2.
Это текст главы 2. Это текст главы
2.<BR>
<A HREF="#up">Наверх</A>
```

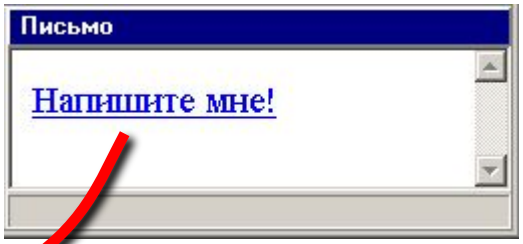
переход на метку

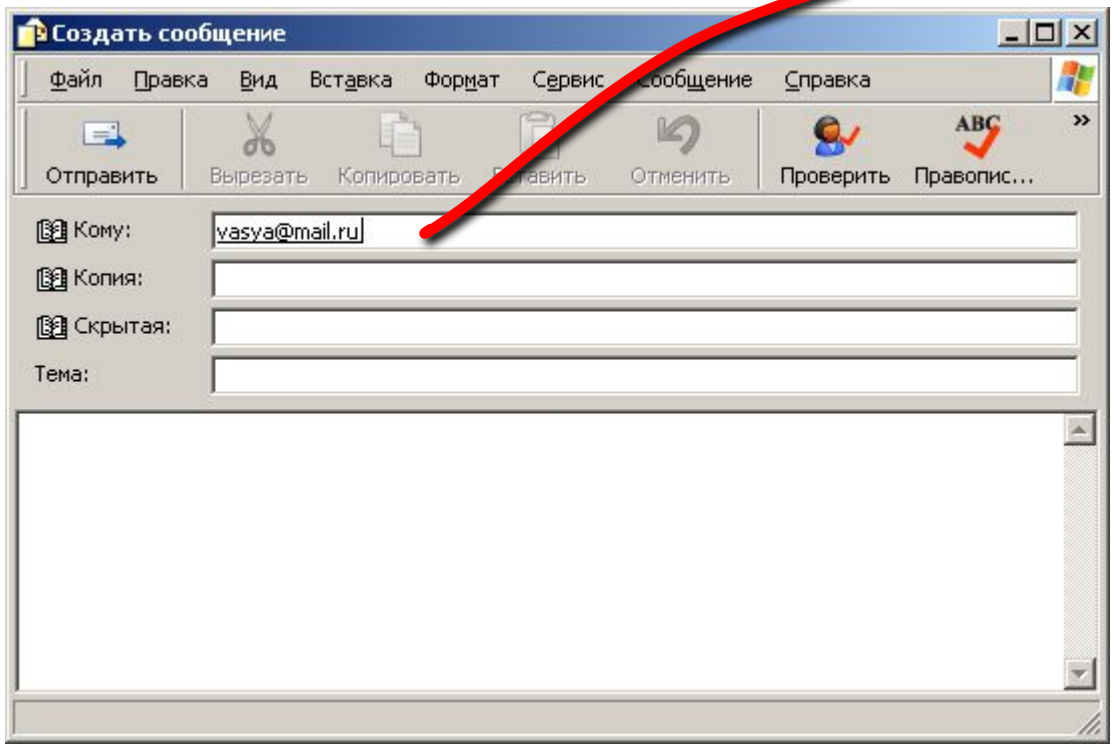
метка (якорь)



Запуск почтовой программы

```
<A HREF="mailto:vasya@mail.ru">
Напишите мне!
</A>
```





Web-страницы. Язык HTML

Тема 6. Этапы создания современного сайта

Этапы создания сайта

Работу по созданию можно разделить на такие этапы:

- Подготовительный;
- Разработка макета;
- Верстка;
- Программирование;
- Наполнение контентом;
- Раскрутка сайта;
- Администрирование (поддержка) сайта.

1. Подготовительный этап

Создание любого сайта должно начинаться с составления его **структуры (карты сайта), тематики, цель сайта и аудитория (пол, возраст) и т.д.**

Пример такой структуры может выглядеть так:

- **Главная**
- **О компании**
  - История
  - Миссия
- **Товары**
  - Автомобили
    - Легковые
    - Внедорожники
  - Мотоциклы
  - Велосипеды
- **Контакты**

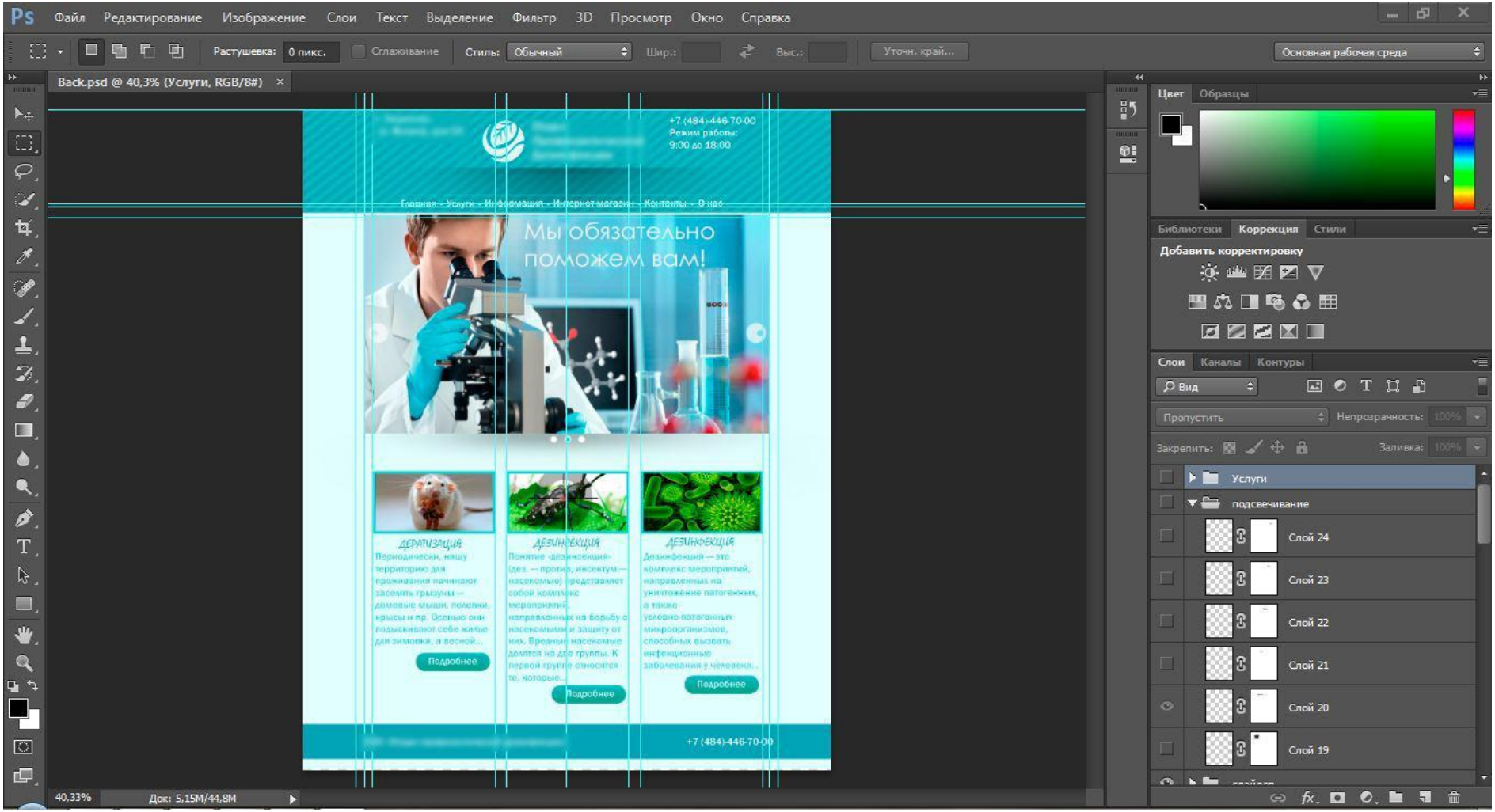
2. Разработка макета

Разработка макета - расположение основных значимых элементов (блоков) на каждой странице (каркас).

Разработка дизайна – отрисовка сайта в графическом редакторе

С помощью языка HTML создаём структуру сайта с использованием блоков и придание им стиля с помощью CSS.

Пример созданного макета в Photoshop



3. Верстка. Виды вёрстки при создании сайтов

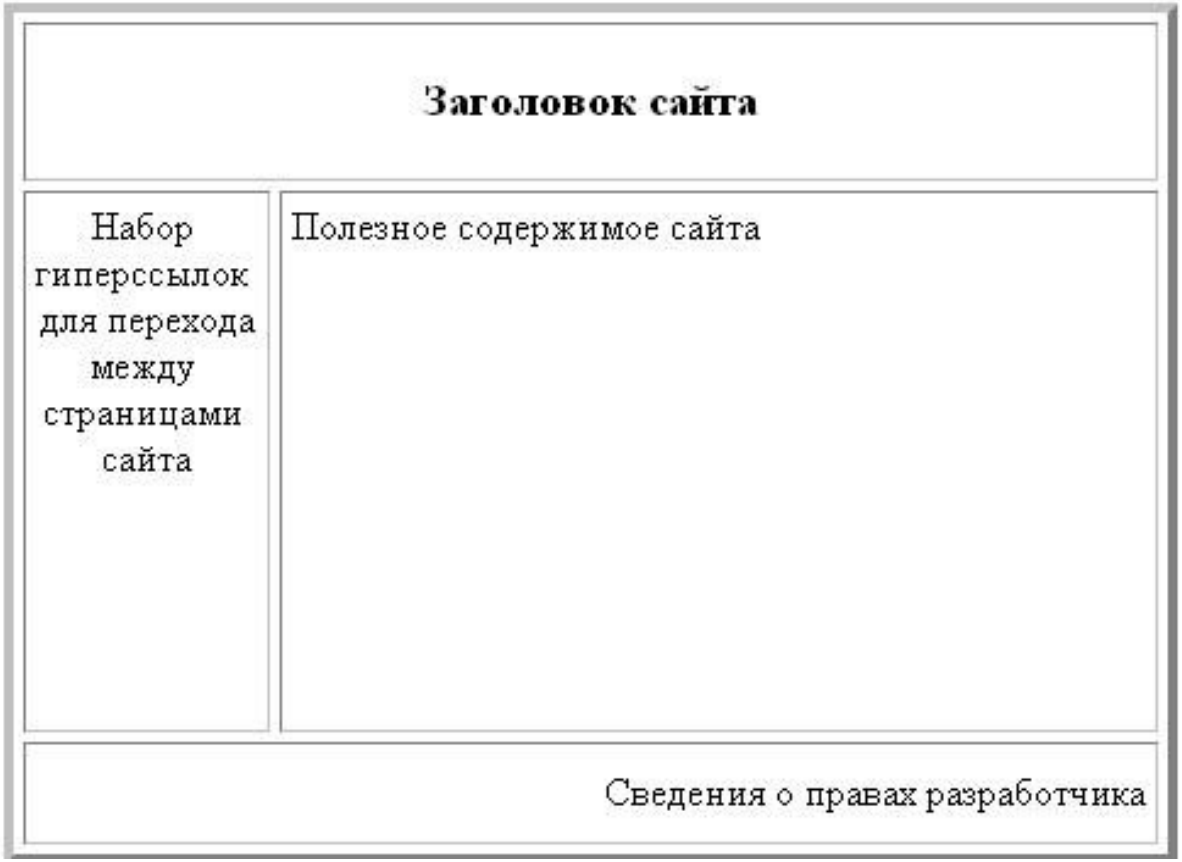
- **Табличная.** В табличной верстке можно создавать колонки, таблицы в браузерах выглядят почти одинаково. Недостатки: индексация такого сайта очень медленная, долго загружаются страницы и код такой верстки слишком большой.

- **Блочная.** Преимущества блочной верстки заключается в том, что она правильно отображает все элементы сайта на разных устройствах (например, на мобильных телефонах). Компактный код, все элементы весят меньше, а значит страница будет загружаться быстро. Из недостатков можно отметить, что при использовании разных браузеров верстка может поплыть.

Табличная верстка <table>

Схема табличного дизайна:

```
<table width="760" border="1" cellspacing="0" cellpadding="5">
<tr> <td colspan="2">Заголовок сайта</td></tr>
<tr> <td>Набор гиперссылок </td><td> Содержимое сайта </td></tr>
<tr> <td colspan="2"> Сведения о правах разработчика </td></tr>
</table>
```



Блочная верстка

Создаем простейший макет сайта на основе блочной верстки.

** **

Элемент ** ** является блочным элементом и предназначен для выделения фрагмента документа с целью изменения вида содержимого. Как правило, вид блока управляется с помощью стилей.

Блоки верстка с

В HTML файле разбиение на блоки

```
<div id="header">Шапка(header)</div>
<div id="left">Левая колонка(left)</div>
<div id="content">Содержание
страницы(content)</div>
<div id="footer">Подвал(footer)</div>
```

В CSS файле придание стиля

```
#header { width:1000px;
          height:217px;
          background: red;
}
#left {     width:200px;
        height:100%;
        float:left;
        background: blue;
}
#content { height:100%;
           width:100%;
           background: green;
}
#footer { width:1000px;
        background: yellow;
}
```

Web-страницы.
Cascading Style Sheets (CSS)

CSS (Cascading Style Sheets)

Каскадные таблицы стилей преимущественно используется как средство описания, оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML.

- используется создателями веб-страниц для задания цветов, шрифтов, расположения отдельных блоков и других аспектов представления внешнего вида этих веб-страниц.
- предоставляет большую гибкость и возможность управления его представлением
- разделение описания логической структуры веб-страницы (которое производится с помощью HTML или других языков разметки) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью формального языка CSS)

Подключение файла со стилем:

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

Синтаксис CSS

Синтаксис CSS состоит из трех частей: селектора, свойства и значения:

Селекторы правила CSS могут быть

- ***селекторами элементов (a, p..)***

p { font-family: Garamond, serif; }

- ***селекторами классов (class)***

.note { color: red;
background: yellow;
font-weight: bold; }

- ***селекторами идентификаторов (id)***

#paragraph1 {
margin: 0;
width:100px;}

- ***селекторами псевдоклассов (a:hover…)***

a:active {
color:yellow;
}

Синтаксис:

селектор {
свойство: значение;
свойство: значение;
свойство: значение;
}

Селекторы элементов

Посмотрим на фрагмент XHTML-документа:

```
<h1>Сказка</h1>
<p>В одной далёкой стране, на краю болота, под пеньком, жил ёжик. И вот
однажды …</p>
```

Из служебной XHTML разметки мы видим только элемент заголовка `h1` и абзаца `p`, и ни слова об оформлении — шрифтах, цвете текста, фоне, отступах и прочем дизайне. Всё это возложено на CSS:

```
/* оформляем заголовки: */
h1 {
  color: red;
  background-color: yellow;
  font: Tahoma 2em;
}
/* оформляем абзацы текста: */
p {
  color: grey;
  line-height: 150%;
}
```

Селекторы классов и идентификаторов

Посмотрим на фрагмент HTML-документа

```
<div id="content">
    <p id="select">Кофе — напиток, изготавливаемый из жареных зёрен кофейного дерева. Благодаря содержанию кофеина оказывает стимулирующее действие.</p>
    <p class="default">До XIV века кофе произрастал в Эфиопии в диком виде.</p>
    <p class="default">Затем, в 1706 году голландские колонисты прислали саженец кофейного дерева в ботанический сад Амстердама, и с этого дерева началось выращивание растения в колониях Нового Света.</p>
</div>
```

В CSS

```
#content {
width: 800px;
background: #ccc;
font-size: 14pt}
```

```
#select {
width: 800px;
font-size: 20pt;
color: blue; }
```

```
.default {
width: 800px;
font-size: 14pt;
}
```

Основные свойста CSS

width:1000px; // задает ширину элемента (можно в px,%)
height:217px; // задает высоту элемента
background: red; // задает цвет фона
background: url('image.jpg'); // задает фоновое изображение
text-align: center; // задает выравнивание текста
float:left; // задает положение блока слева
a:hover // изменяет стильссылки при наведение курсора
a:visited // изменяет стиль посещенной ссылки

margin: 0 auto; // задает отступ сверху/снизу справа/слева
padding: 20px 20px; // задает отступ внутри блока

Остальные свойства можно посмотреть на сайтах:
http://css.manual.ru
https://webref.ru/https://webref.ru/ (http://htmlbook.ru)

CSS 3. Новые возможности

- **Прозрачность/Opacity**

В браузерах, поддерживающих это свойство, указать прозрачность можно так:

В файле стилей нужному селектору указать следующие свойства:

```
background-color: rgb(0,0,255);
opacity: 0.5;
```

- **Указание нескольких фоновых картинок/Multiple Backgrounds**

Новая версия CSS позволяет добавлять элементам несколько фоновых картинок. Можно разместить изображения вверху, в центре, в углу и в других местах. Верстать сложные макеты станет намного проще.

Вот пример установки нескольких картинок для фона:

В файле стилей нужному селектору указать следующие свойства:

```
background: url(body-top.gif) top left no-repeat,
url(banner_fresco.jpg) top 11px no-repeat,
url(body-bottom.gif) bottom left no-repeat,
url(body-middle.gif) left repeat-y;
```

CSS 3. Новые возможности

- **Пользовательское изменение размера/Resize**

Средствами новой версии CSS можно добавить немного интерактивности на ваш сайт - свойство resize позволит посетителю изменять размер э

```
div.resize { width: 100px; height: 100px; resize: both; overflow: auto; }
```

- **Закругленные углы/border-radius**

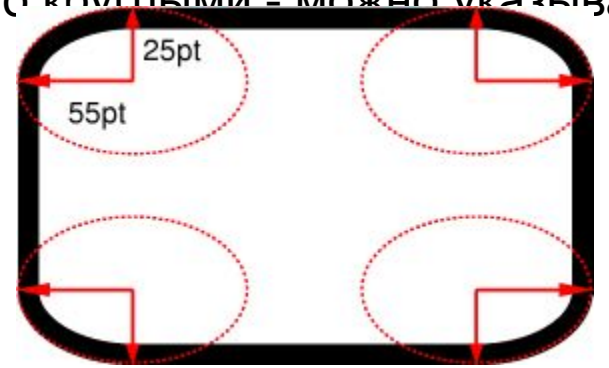
С помощью этого свойства возможно делать закругленные углы у блоков.

```
#elem { border-top-left-radius: 1em; border-top-right-radius: 2em;
border-bottom-right-radius: 3em; border-bottom-left-radius: 4em; }
```

ти.

Границы могут быть не только идеально круглыми - можно указывать два

```
border-radius: 55pt 25pt;
```



CSS 3. Новые возможности

- **Тень блока/box-shadow**

Абсолютно новое свойство, позволяющее показать бокс с тенью. Его формат таков:

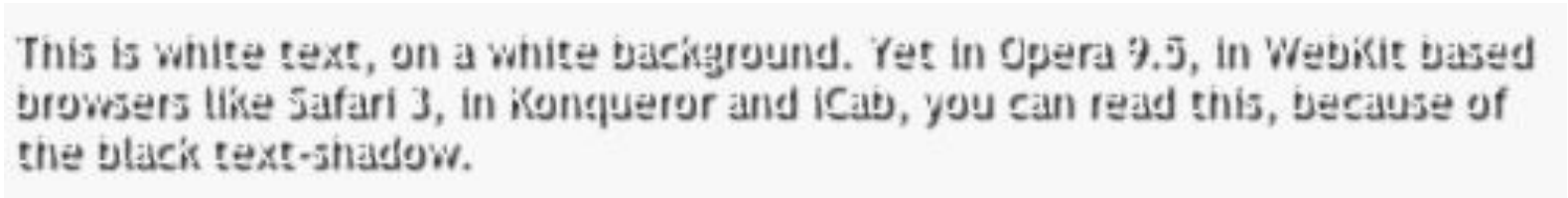
```
span { box-shadow: 0.2em 0.2em 5px #CCC; }
```

Результат можно увидеть на картинке: 

- **Текст с тенью/text-shadow**

Это свойство не совсем новое, оно присутствовало и в CSS2, однако поддерживают его пока только браузеры Opera 9.5, Safari 3, Konqueror и iCab. Если ИЕ8 и Firefox 3 не отстанут, применять Photoshop для того, чтобы сделать простую тень, станет не нужно.

```
color: #fff; background-color: #fff; text-shadow: 2px 2px 2px #000;
```



- **Добавление простой анимации**

Подробно ознакомиться можно здесь: http://alt-f4.ru/article/new-in-css3

Web-страницы. JavaScript

Немного о JavaScript

Язык JavaScript предоставляет средства для решения многих задач в документе HTML без необходимости взаимодействия с сервером.

Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Например, выпадающее или раскрывающее меню, слайд-шоу из изображений, проверка правильности ввода данных в форму, отображение изображений в новом окне с эффектами и т.д.

Вставка кода в HTML-документ в контейнерах <script> ... </script>

Объектная модель DOM

В javascript страница представлена в виде объектной модели **DOM (Document Object Model).**

Согласно DOM-модели, документ является иерархией. Каждый HTML-тег образует отдельный элемент-узел, каждый фрагмент текста - текстовый элемент, и т.п.

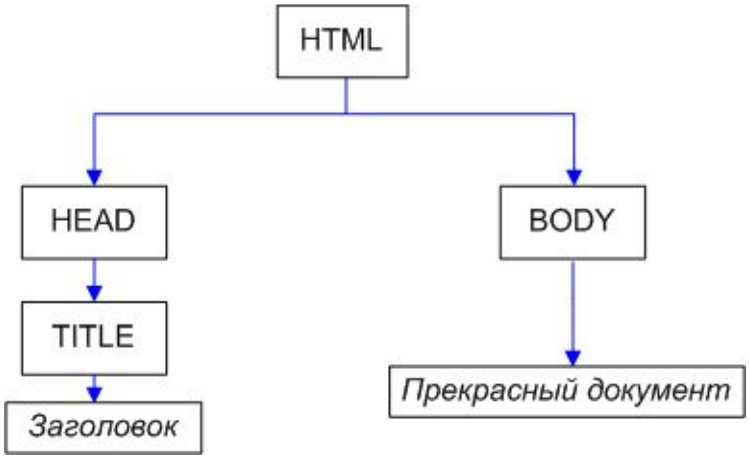
Проще говоря, DOM - это представление документа в виде дерева тегов. Это дерево образуется за счет вложенной структуры тегов плюс текстовые фрагменты страницы, каждый из которых образует отдельный узел.

Простейший DOM

Построим, для начала, дерево DOM для следующего документа.

```
1  <html>
2    <head>
3      <title>Заголовок</title>
4    </head>
5    <body>
6       Прекрасный документ
7     </body>
8  </html>
```

Самый внешний тег - <html>, поэтому дерево начинает расти от него.
Внутри <html> находятся два узла: <head> и <body> - они становятся дочерними узлами для <html>.



Теги образуют *узлы-элементы* (element node). Текст представлен *текстовыми узлами*(text node). И то и другое - равноправные узлы дерева DOM.

Обращение к элементам DOM

Стандарт DOM предусматривает несколько средств поиска элемента.

Это методы: **getElementById, getElementsByTagName и getElementsByName.**

Самый удобный способ найти элемент в DOM - это получить его по id. Для этого используется вызов **document.getElementById(id)**

Например, следующий код изменит цвет текста на голубой в div'e с id="dataKeeper":

```
document.getElementById('dataKeeper').style.color = 'blue'
```

Обращение к элементам DOM

Следующий способ - это получить все элементы с определенным тегом, и среди них искать нужный. Для этого служит **document.getElementsByTagName(tag)**.

Она возвращает массив из элементов, имеющих такой тег. Например, можно получить второй элемент(нумерация в массиве идет с нуля) с тэгом li:

getElementsByTagName можно вызывать не только для document, но и вообще для любого элемента, у которого есть тег (не текстового).
При этом будут найдены только те объекты, которые находятся под этим элементом.
Например, следующий вызов получает список элементов LI, находящихся внутри первого тега div:

```
document.getElementsByTagName('DIV')[0].getElementsByTagName('LI')
```

Возможности, которые даёт DOM

Каждый DOM-элемент является объектом и предоставляет свойства для манипуляции своим содержимым, для доступа к родителям и потомкам.

Пример создания раскрывающего меню:

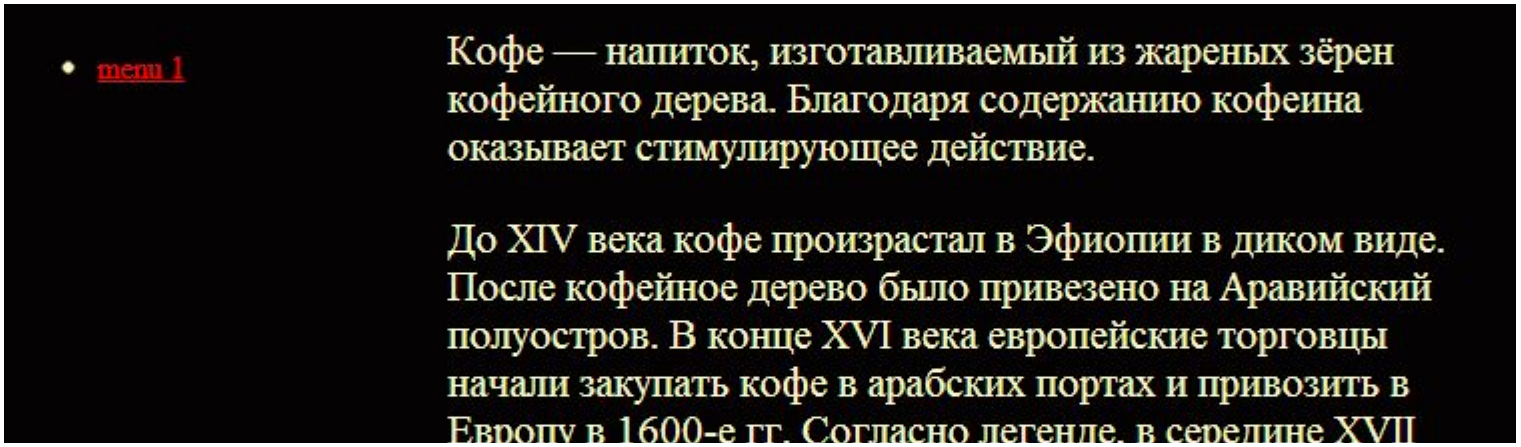
```
<ul id="menu">
        <li><a href="#" onclick="openMenu(this);return false">menu 1</a>
    <ul>
            <li><a href="index.html">Главная</a></li>
            <li><<a href="history.html">История кофе</a></li>
            <li><<a href="vir.html">Выращивание</a></li>
            <li><<a href="klasif.html">Классификация</a></li>
            <li><<a href="forma.html">Написать нам</a></li>
    </ul>
        </li>
</ul>
```

Вставить код JavaScript на страницу

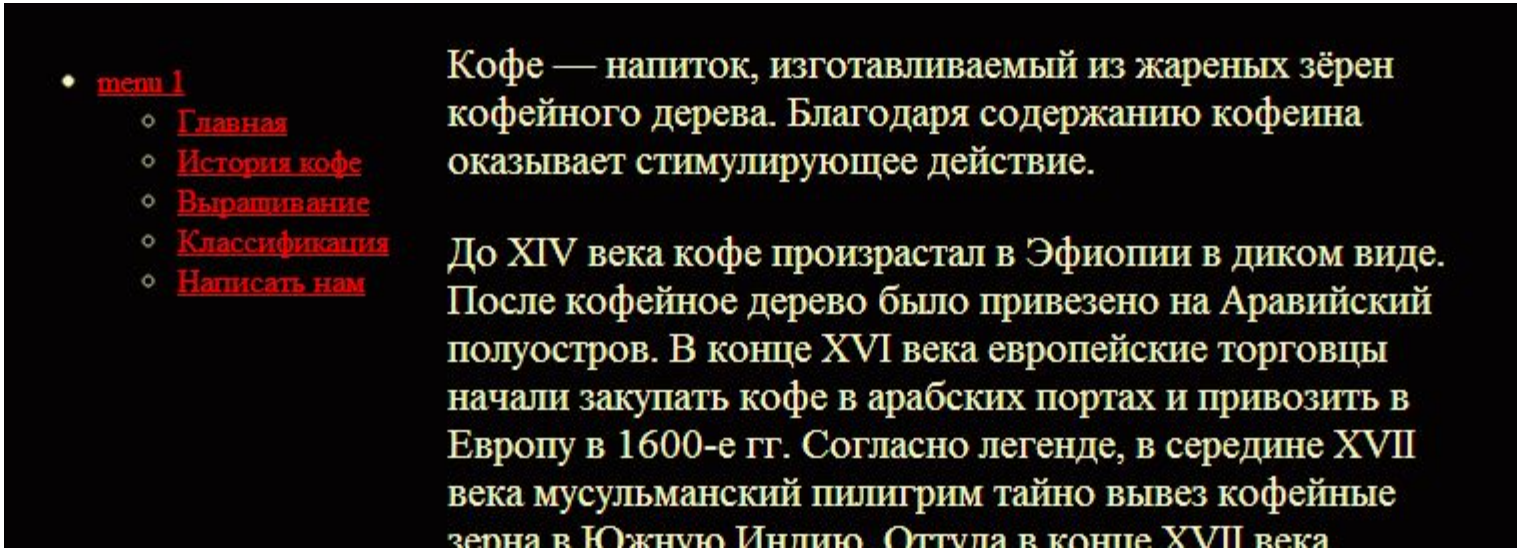
Вставляем следующую функцию JavaScript в любом месте веб-страницы с меню:

```
<script>
function openMenu(node){
    var subMenu =
node.parentNode.getElementsByTagName("ul")[0];
    subMenu.style.display == "none" ?
subMenu.style.display = "block" : subMenu.style.display =
"none";
}
</script>
```

Результат



При нажатии:



jQuery

jQuery — библиотека JavaScript, фокусирующаяся на взаимодействии JavaScript и HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими.

jQuery. Подключение

Подключить jQuery можно двумя способами:

1.Загрузка и подключение

Загружать jQuery желательно с http://jquery.com/download/. Чтобы закачать файлы по этим ссылкам, нужно кликнуть по ним правой клавишей и выбрать "Сохранить ссылку как".

После того как вы скачали нужный файл с jQuery, нужно закачать его на сервер, где располагается ваш сайт и подключить его на страницы своего сайта, прописав до него путь:

```
<!DOCTYPE html>
<html>
<head>
  <script src="/js/jquery.min.js"></script>
  ...
</head>
<body>
  ...
</body>
</html>
```

jQuery. Подключение

2. Подключение jQuery с CDN (не закачивая библиотеку на сервер)

Существуют несколько таких хранилищ, наиболее известные и надежные из них

- Google CDN (https://developers.google.com/speed/libraries/?hl=ru-RU&csw=1#jquery),
- Microsoft CDN (http://www.asp.net/ajax/cdn),
- CDN который организовали создатели jQuery(http://code.jquery.com).

Подключить jQuery с CDN не сложнее, чем со своего сервера — необходимо просто прописать соответствующий путь до него. Вот например как подключается библиотека с официального CDN jQuery:

```
<!DOCTYPE html>
<html>
<head>
  <script src="http://code.jquery.com/jquery-1.8.3.js"></script>
  ...
</head>
<body>
  ...
</body>
</html>
```

jQuery. Пример(слайдер)

jQuery можно использовать для различных задач.
Разберем например слайдер и галерею изображений.

Разберем самый простой слайдер, который можно взять здесь:
https://github.com/k-ivan/jQuery-simple-slider/archive/master.zip

- Скачаем файлы слайдера и распакуем их на сервер.
- Подключим файл стиля слайдера slider.css (в нем можно менять стиль слайдера), библиотеки jQuery и самого слайдера

Внутри тега head прописываем

```
<link rel="stylesheet" href="slider.css">

<script src="//ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>

<script src="slider.js"></script>
```

jQuery. Пример(слайдер)

- **Добавляем на страницу следующий код**

```
<div class="slider-container">
  <div class="slider" id="slider">
    <div class="slide">
      <img src="img-1.jpg" alt="">
      <span class="caption"><strong>Текст 1<a href="">Далее</a> </span>
    </div>
    <div class="slide">
      <img src="img-2.jpg" alt="">
      <span class="caption">текст 2</span>
    </div>
    <div class="slide">
      <img src="img-3.jpg" alt="">
      <span class="caption">Текст 3</span>
    </div>
  </div>
    <a href="" class="switch" id="prev"><span></span></a>
    <a href="" class="switch" id="next"><span></span></a>
</div>
```

jQuery. Пример(слайдер)

Как изменять слайдер:

1) Чтобы добавить(удалить) страничку слайдера добавляем(удаляем) элемент

```
<div class="slide">
    <img src="img-1.jpg" alt="">
    <span class="caption"><strong>Текст 1<a href="">Далее</a> </span>
</div>
```

2) Изменение картинки :
В теге Img в атрибуте src указываем ссылку на нужное изображение.

3)Изменение текста на слайде
В теге span (текст 2) меняем текст на нужный.

jQuery. Пример(слайдер)

- Вставляем Инициализацию плагина

1. Без параметров
<script> $(".slider-container").sliderUi(); </script>

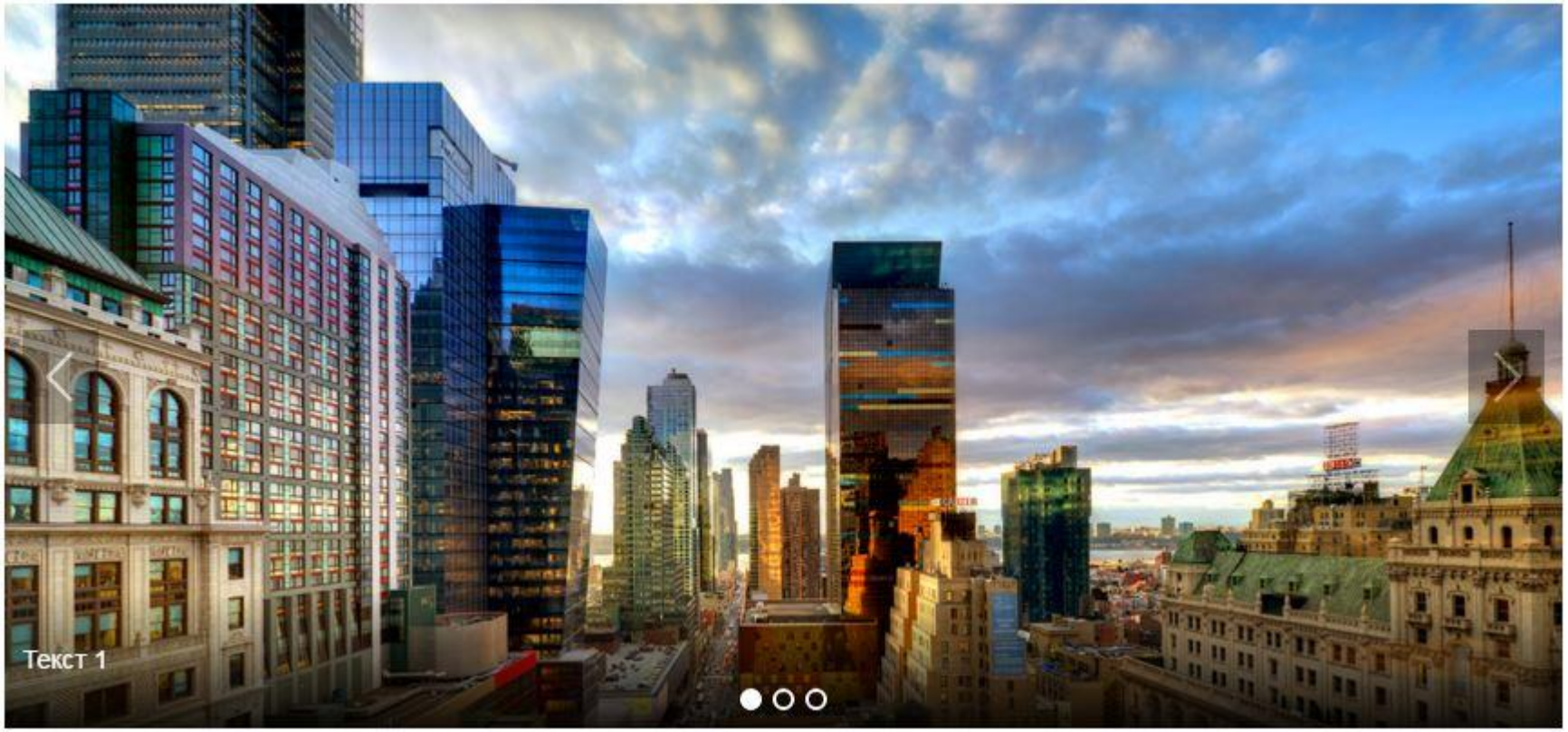
2. С расширенными параметрами

```
<script>
    $(".slider-container").sliderUi({
       speed: 500,
       cssEasing: "ease-in-out" //описание easing для css анимации
    });
</script>
```

Список параметров, которые можно изменять:
controlShow: true, // показывать навигацию внизу
arrowsShow: true, // показывать вперед|назад навигацию
autoPlay: true, // автоматическое перелистывание изображений
delay: 3000, // задержка перед перелистыванием
caption: false, // показывать описание
speed: 300, // скорость анимации
cssEasing: "ease-out" // функция плавности анимации с помощью CSS

jQuery. Пример(слайдер)

Результат:



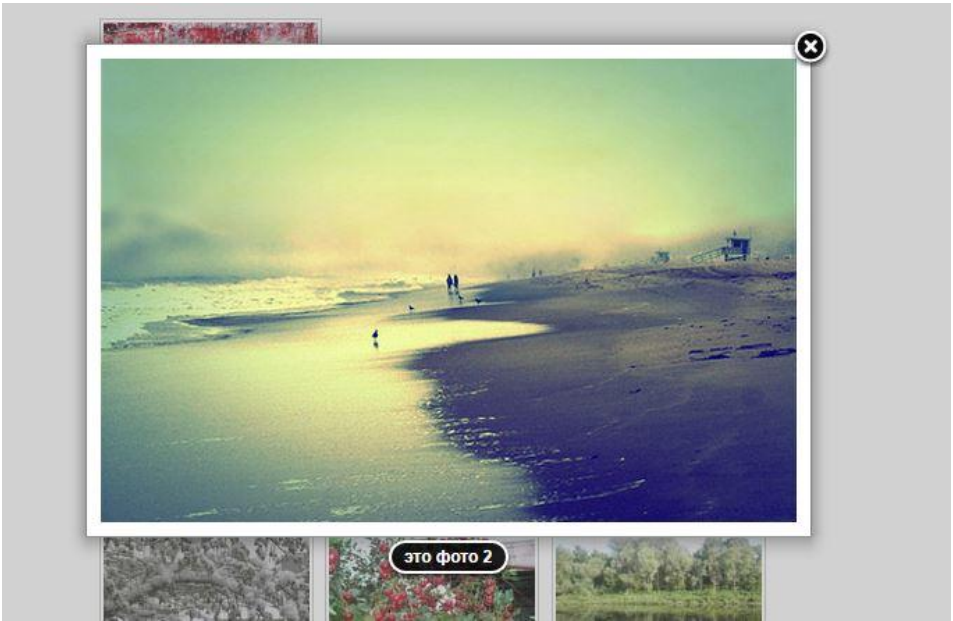
jQuery. Пример(галерея изображений)

Использование jQuery для создания галереи изображений

Галерея **Funcy Box** (подробную инструкцию по установке плагина и исходники можно найти здесь http://ruseller.com/lessons.php?id=238&rub32)

Общий вид

При нажатии на картинку



jQuery UI.

jQuery UI (User intarface)— библиотека JavaScript с открытым исходным кодом для создания насыщенного пользовательского интерфейса в веб-приложениях, часть проекта jQuery. Построена поверх главной библиотеки jQuery и предоставляет разработчику упрощенный доступ к её функциям взаимодействия, анимации и эффектов, а также набор виджетов.

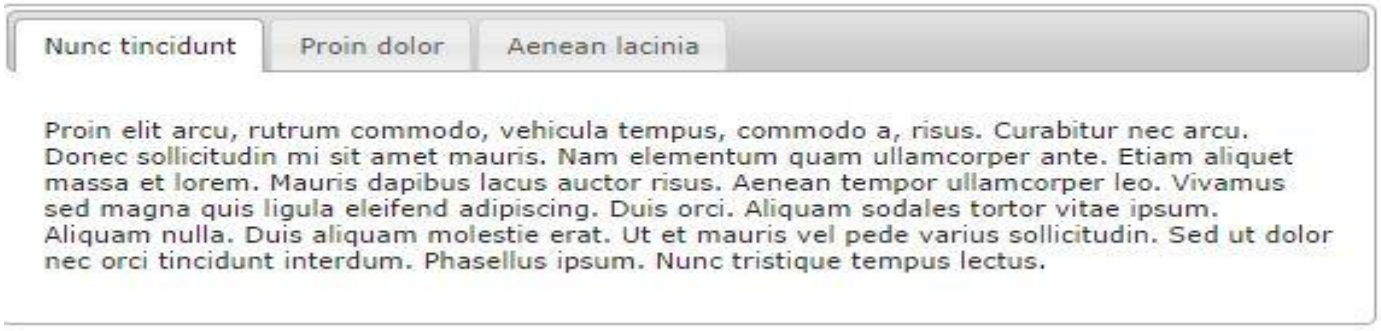
jQuery UI. Datapicker

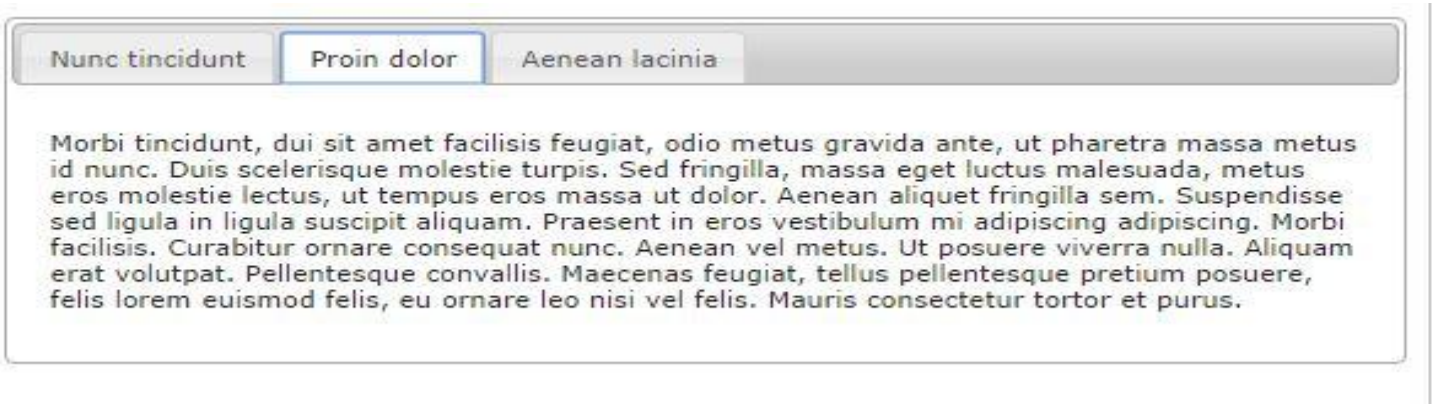
Datepicker — виджет для выбора даты или диапазона дат


```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery UI Datepicker - Default
functionality</title>
<link rel="stylesheet"
href="//code.jquery.com/ui/1.11.4/themes/smoothness/j
query-ui.css">
<script
src="//code.jquery.com/jquery-1.10.2.js"></script>
<script
src="//code.jquery.com/ui/1.11.4/jquery-ui.js"></scri
pt>
<link rel="stylesheet"
href="/resources/demos/style.css">
<script>
$(function() {
$( "#datepicker" ).datepicker();
});
</script>
</head>
<body>
<p>Date: <input type="text" id="datepicker"></p>
```

jQuery UI. Tabs

Tabs — вкладки



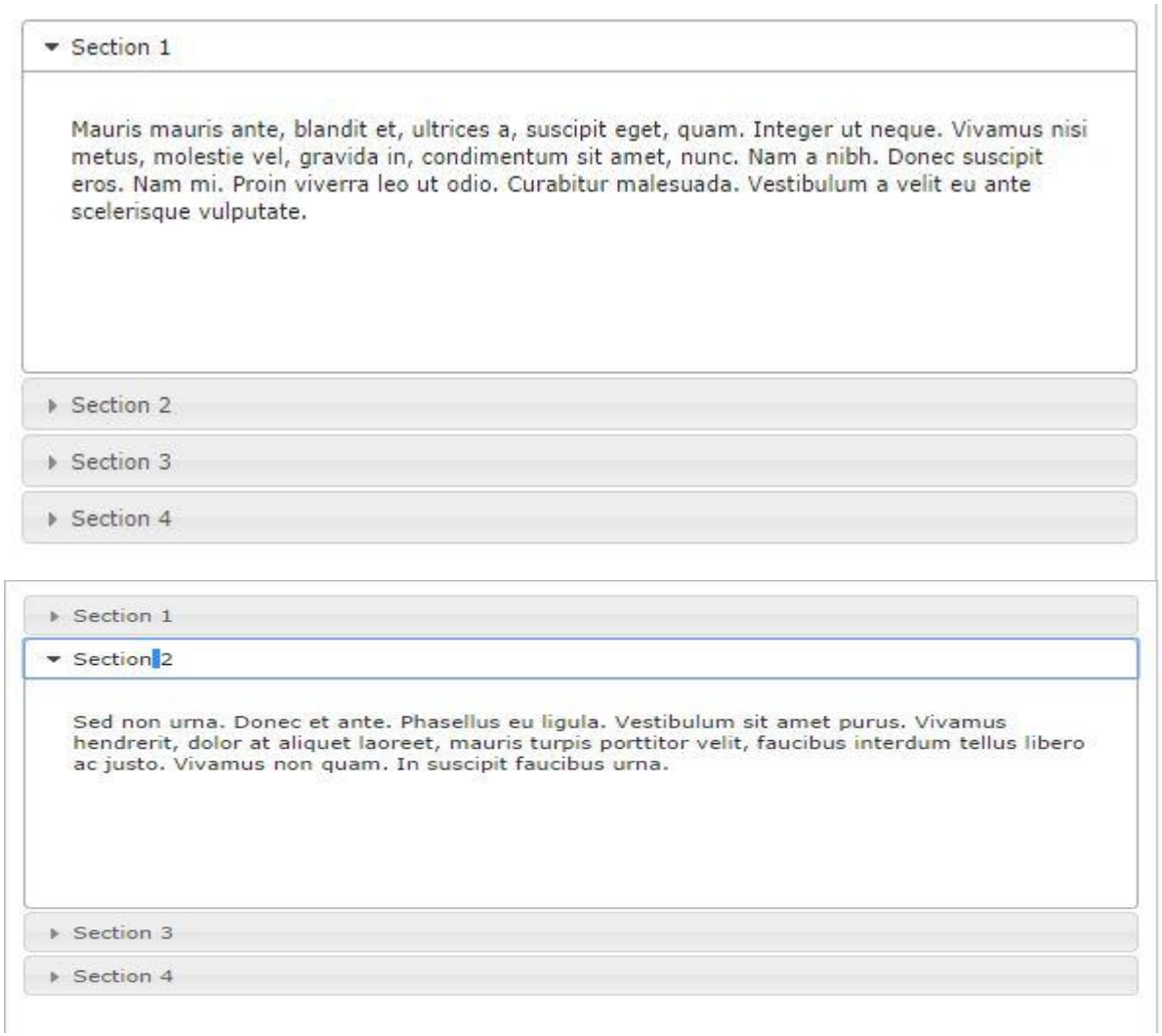


jQuery UI. Tabs

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery UI Tabs - Default functionality</title>
<link rel="stylesheet"
href="//code.jquery.com/ui/1.11.4/themes/smoothness/jquery-ui.css">
<script src="//code.jquery.com/jquery-1.10.2.js"></script>
<script src="//code.jquery.com/ui/1.11.4/jquery-ui.js"></script>
<link rel="stylesheet" href="/resources/demos/style.css">
<script>
$(function() {
$( "#tabs" ).tabs();
});
</script>
</head>
<body>
<div id="tabs">
<ul>
<li><a href="#tabs-1">Название 1 вкладки</a></li>
<li><a href="#tabs-2"> Название 2вкладки </a></li>
<li><a href="#tabs-3"> Название 3 вкладки </a></li>
</ul>
<div id="tabs-1">
<p> ест первой вкладки</p>
</div>
<div id="tabs-2">
<p> Текст 2 вкладки.</p>
</div>
<div id="tabs-3">
<p> Текст 3 вкладки</p>
</div>
</div>
</body>
</html>
```

jQuery UI. Accordion

Accordion — виджет «Аккордеон»



jQuery UI. Accordion

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery UI Accordion - Default functionality</title>
<link rel="stylesheet"
href="//code.jquery.com/ui/1.11.4/themes/smoothness/jquery-ui.css">
<script src="//code.jquery.com/jquery-1.10.2.js"></script>
<script src="//code.jquery.com/ui/1.11.4/jquery-ui.js"></script>
<link rel="stylesheet" href="/resources/demos/style.css">
<script>
$(function() {
$( "#accordion" ).accordion();
});
</script>
</head>
<body>
<div id="accordion">
<h3>Секция 1</h3>
<div>
<p>
текст
</p>
</div>
<h3> Секция 2</h3>
<div>
<p>
текст
</p>
</div>

</div>
</body>
</html>
```

Landing Page

LANDING PAGE - веб-страница, построенная определенным образом, основной задачей которой является сбор контактных данных целевой аудитории.

Landing Page также часто называют одностраничником. Так как вся информация и все действия осуществляются с помощью одной страницы.

Особенности Landing Page:

- только **одна**, хорошо запланированная **цель:** покупка товара, заполнение формы, скачивание ссылка.
- **призыв к действию** (call to action), напр. Заполнить форму, Купить
- **простой** дизайн
- короткие и связные тексты, написанные на **языке выгоды**, отстуствие отвлекающих элементов
- **отсутствие** классического **меню**.

Landing Page. Шаблон

ДИСКРИПТ

Заголовок или слоган, характеризующий вашу нишу

ИЗОБРАЖЕНИЕ

Фотография или рисунок товара/услуги, который получит посетитель

ТРИГГЕРЫ ДОВЕРИЯ

Отзывы клиентов, список клиентов, успешные кейсы

СХЕМА РАБОТЫ

Доступная и понятная схема вашей работы

ПРИЗЫВ

Побуждающая к действию фраза или иконка

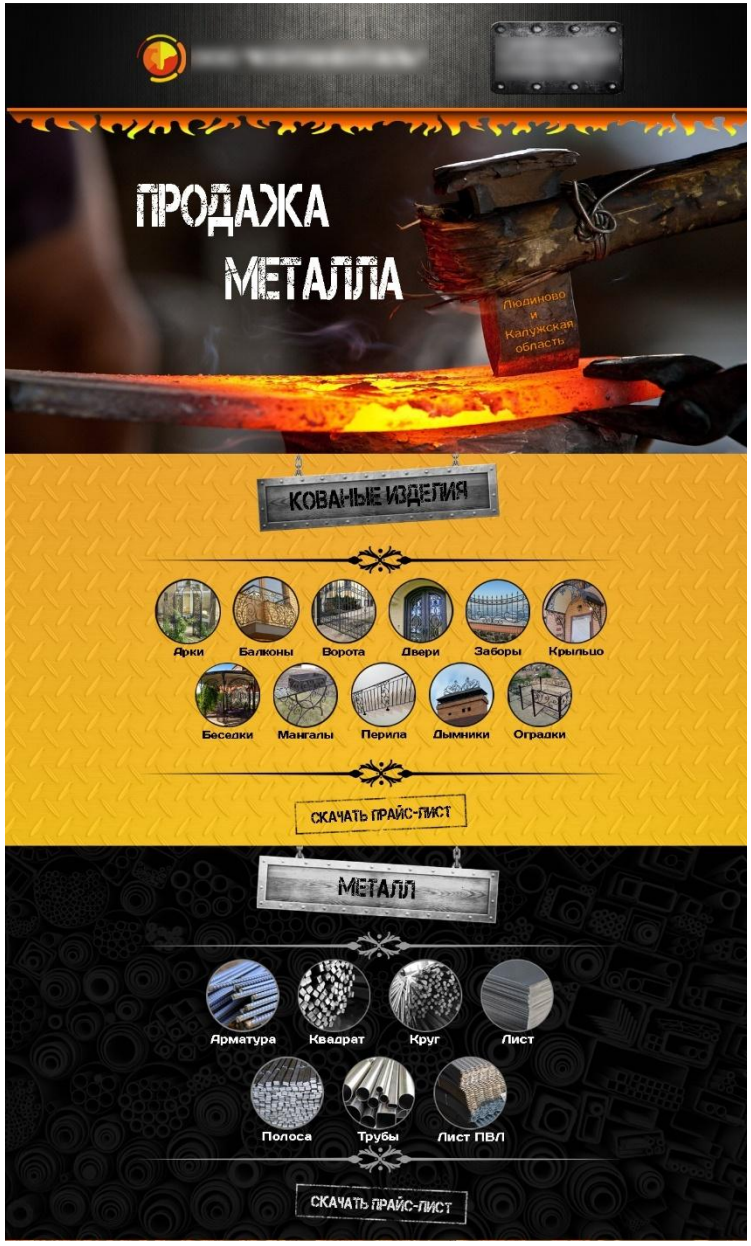
КНОПКА ЗАКАЗА

Большая, хорошо выделенная кнопка

ЗАКРЫТИЕ ВОЗРАЖЕНИЙ

Примеры работ, ответы на вопросы, бонусы, акции и счетчики обратного отсчета

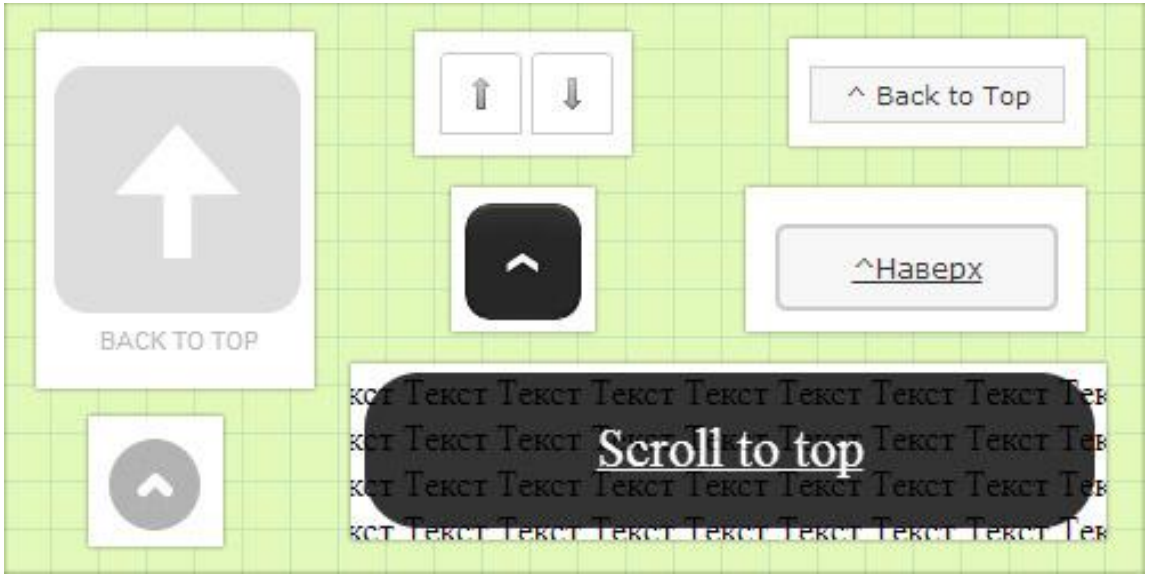
Landing Page. Пример



Landing Page. Перемещение по странице

Когда на странице сайта расположено большое количество контента пользователь ознакомившись с ним и промотав страницу до определенного момента или до самого низа, часто начинает скролить ее верх, чтобы увидеть навигационные ссылки расположенные в верхней части страницы и\или произвести какие-то действия.

Чтобы добавить удобства пользователям и в целом улучшить юзабилити своего сайта лучше всего использовать для сайта кнопку вверх, нажав на которую пользователь автоматически перенаправляется в верхнюю часть страницы.



Landing Page. **Кнопка наверх с помощью jQuery**

Кнопка реализована достаточно просто. Для ее функционирования понадобится библиотека jQuery, небольшой скрипт содержащий события jQuery, определенные стили и тег DIV содержащий сам текст и необходимый ID.

Представленный ниже код вам необходимо вставить переде тегом </head> на всех страницах вашего сайта. Если библиотека jQuery уже подключена к вашему сайту, то первую строчку добавлять не нужно.

```
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js"></script>
 <script type="text/javascript">
 $(function() {
 $(window).scroll(function() {
 if($(this).scrollTop() != 0) {
 $('#toTop').fadeIn();
 } else {
 $('#toTop').fadeOut();
 }
 });
 $('#toTop').click(function() {
 $('body,html').animate({scrollTop:0},800);
 });
 });
 </script>
```

Landing Page. Кнопка наверх.

Чтобы придать красивый внешний вид кнопке необходимо добавить следующие стили. Их лучше всего добавить в файл стилей вашего сайта.

```
#toTop {
width:100px;
border:1px solid #ccc;
background:#f7f7f7;
text-align:center;
padding:5px;
position:fixed;
bottom:10px; /* отступ кнопки от
нижнего края страницы*/
right:10px;
cursor:pointer;
display:none;
color:#333;
font-family:verdana;
font-size:11px;
}
```

Для вызова кнопки добавьте следующий HTML код перед тегом </body>.

```
<DIV ID = "toTop" > ^ Наверх </ DIV >
```

Web-страницы. PHP+MySQL

Обработка событий

Динамика на веб-страницах реализована при помощи скриптов, которые выполняются на сервере.

Работает это следующим образом:

1. Браузер запрашивает у сервера документ;
2. Сервер определяет, что документ является скриптом и запускает его на выполнение;
3. Скрипт выполняется (генерирует html страницу);
4. Сервер отправляет сгенерированную страницу браузеру.

Существует несколько языков программирования, на которых могут писать скрипты, которые генерируют "динамические" страницы. Самые распространенные из них:

Perl SSI PHP ASP Python Java

Схема работы без и с PHP

Статические HTML-страницы

Динамические страницы (с PHP кодом)

Основы PHP

PHP выполняется на сервере. Браузер посылает серверу запрос на страницу с php кодом. Сервер отдает эту страницу на исполнение интерпретатору PHP, интерпретатор генерирует HTML код, отдает серверу, а сервер посылает клиенту.

Никакого PHP кода в браузер не попадает (**это важно! Это значит, что увидеть исходный код PHP скрипта невозможно!**). Единственный способ отправить что-то скрипту - это кликнуть по ссылке или нажать на кнопку в форме.

<? скрипт ; ?>

либо

<?php скрипт ; ?>

// - не выполняются команды от данных символов до конца строки;
/ комментарий */* - не воспринимаются команды между данными символами независимо от количества строк комментария;

Что необходимо

- **Apache HTTP-сервер**

необходим для обработки запросов от браузера и передачи на исполнение php-скриптов (т.к. браузер не выполняет php-скрипт).

- **Интерпретатор PHP**

выполнение php-скрипта

- **MySQL (необязательно)**

если собираемся использовать Базу данных (или можно сохранять данные в файл)

!Чтобы не устанавливать всё отдельно можно воспользоваться Джентльменским набором Web-разработчика

(«Д.н.w.р», читается «Денвер»)

www.denwer.ru

Денвер

Локальный сервер (Apache, PHP, MySQL, Perl и т.д.) и программная оболочка, используемые Web-разработчиками для разработки сайтов на «домашней» (локальной) Windows-машине без необходимости выхода в Интернет.

По умолчанию Денвер устанавливается в папку C:/WebServers

Для дальнейшей работы необходимо создать в папке **home** папку с вашим доменным именем, и в ней папку www.

Например, если доменное имя praktika.ru, то C:/WebServers/home/praktika.ru/www

Далее скопировать все ваши файлы с эту папку, переименовать главный файл в index.html и запустить Денвер (файл Run.exe в папке denwer)

PHP в действии

Затем открываем браузер и переходим по адресу **praktika.ru**.
Должен открыться Ваш сайт (т.к. этот запрос обрабатывает Денвер)
Пока вы увидите просто ваши HTML-страницы

Для того, чтобы сообщить серверу о том, что надо произвести обработку PHP-кода, необходимо использовать следующий синтаксис при добавлении PHP в HTML-документ:

<?php ...здесь идет PHP-код ?>

Открытие блока PHP-кода обозначается как "<?php", а закрытие - "?>".
Теперь попробуйте вставить в любое место код следующим образом:

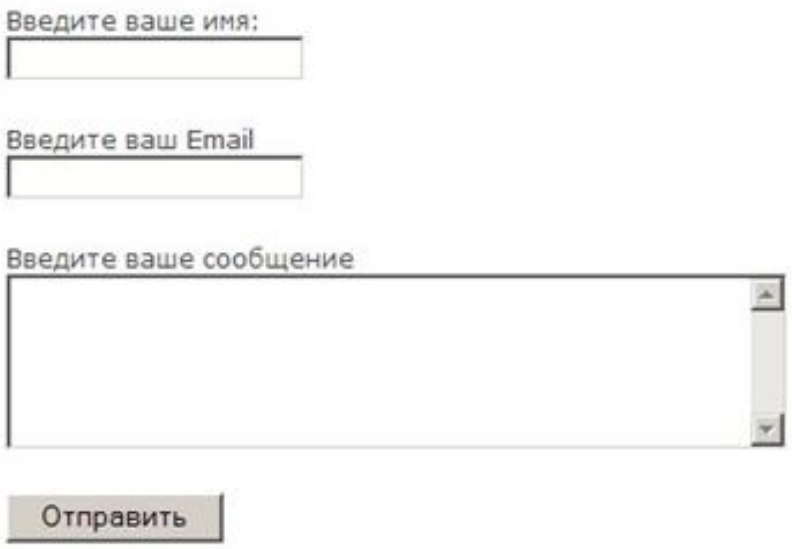
<?php echo "Это PHP в действии"; ?>

А потом можете просмотреть в браузере исходный текст полученной страницы. Никаких тегов PHP там нет! Только текст **Это PHP в действии.**

Потому, что PHP исполняется на сервере!

Форма HTML

Форма — это инструмент, с помощью которого HTML-документ может послать некоторую информацию скрипту, где информация будет некоторым образом обработана.



Форма открывается тэгом ***<FORM>*** и заканчивается тэгом ***</FORM>***.
Для того, чтобы запустить процесс передачи данных из формы обработчику, нужен какой-то орган управления. Создать такой орган управления очень просто:

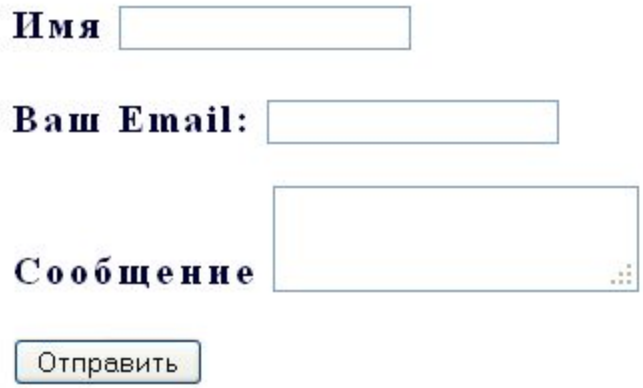
<input type="submit" value="Отправить">

браузер нарисует на экране кнопку с надписью *Отправить* , при нажатии на которую все имеющиеся в форме данные будут переданы обработчику, определенному в метке <FORM action="">.

Пример формы

И так создадим простейшую форму:

<form name="form1" method="post" action="test.php"> (начало формы)
* **Имя:** <input type="text" name="sirname"> (поле имени) *
* **Ваш Email:** <input type="text" name="email"> *
* **Сообщение***
<textarea name="message"></textarea> (поле сообщения)
* *
* *
<input type="submit" name="send" value="Отправить"> (кнопка отправки)
* *
</form> (конец формы)



! В свойстве action тега Form указывается файл скрипта имя_файла.php

Передача данных на сервер

Тэг **<form>**, имеющий парный завершающий тэг **</form>**, собственно и задает форму. Его атрибуты - оба необязательные:

- **action** - указывает URL (полный или относительный), на который будет отправлена форма. Отправка формы - это такой же запрос к серверу, как и все остальные (как я уже описал выше). Если этот атрибут не указать - форма отправляется на текущий документ, то есть "сама на себя".
- **method** - способ отправки формы. Их два:

  GET - отправка данных формы в адресной строке. Вы могли заметить на различных сайтах присутствие в конце URL символа "?" и следующих за ним данных в формате параметр=значение. Здесь "параметр" соответствует значению атрибута name элементов формы (см. ниже про тэг <input>).

  POST - данные формы отправляются в теле запроса. Если не совсем понятно (или совсем непонятно), что это такое - не беспокойтесь, скоро мы к этому вопросу вернемся.

Передача данных методом GET

GET - это название запроса который отправляется на сервер скрипту с помощью браузера открыто, через URL, адресную строку.

Если в адресной строке вы увидели знак амперсанды (&) и знак вопрос (?), можно считать, что этот узел работает на PHP, и ему в данный момент отправлены переменные и их значения.
Выглядеть это может примерно так:
http://lphp.ru/index.php?page=4&id_artpage=43

http:// - префикс основного протокола передачи данных в web (HTTP)
lphp.ru - домен в котором находится сервер
index.php - имя файла, который будет обрабатывать запрос, то есть кому собственно отправлен GET-запрос
? - разделитель, после которого перечисляются переменные со значениями, которые нужны скрипту обработчику для формирования ответной страницы
page - переменная или имя переменной
= - оператор присваивает значения переменной
4 - значение переменной
& - разделитель в строке запроса, между парами (переменная=значение**&**переменная=значение) и т.д.

Передача данных методом POST

Главное отличия метода POST от метода GET это то, **что он скрывает все передаваемые им переменные и их значения, в своём теле**.

При передачи методом **POST** значения помещаются на сервере в глобальный массив **$_POST[‘имя элемента’].**

При передачи методом **GET** в глобальный массив **$_GET[‘имя элемента’].**

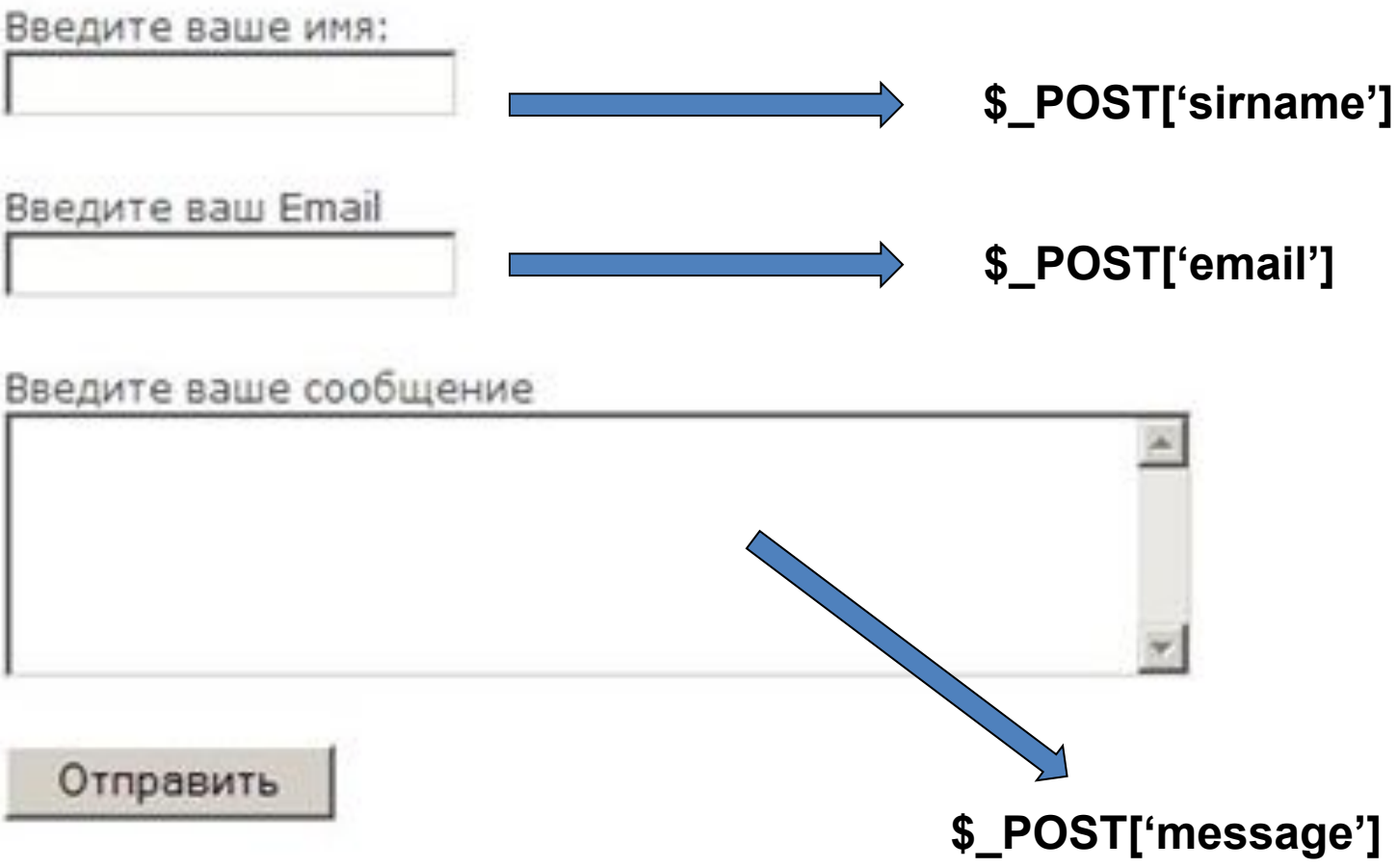
где имя элемента указано в атрибуте **name** соответствующего элемента формы, например,

<input type=“text” name="sirname">

или

<textarea name="message"></textarea>

Передача значений



Немного о языке PHP

echo - выводит одну или более строк
$имя_переменной = значение; *//определение переменной*

Примеры использования echo

```
<?php
echo "Привет мир!";
echo "Это займет
несколько строк. Переводы строки тоже
выводятся";

// с echo можно использовать переменные ...
$foo = "foobar";
$bar = "barbaz";
echo "foo - это $foo"; // foo - это foobar

// с echo можно выводить значение глобальных массивов
echo $_POST[‘name’];
?>
```

Обработка переданных значений

А теперь – создаем файл **test.php и пишем скрипт**:

```
<?
echo "<h1>Привет, <b>".$_POST['sirname']."</b></h1>!";
echo "Email:<b>".$_POST['email']."</b>";
echo "Сообщение: ".$_POST['message'];
?>
```

Заполняем форму и смотрим результат.
Должны вывестись заполненные в форме данные на странице после обработки скрипта.

Хранение данных

Хранилище обработанных данных

Текстовый файл (.txt)

- простота использования
- не зависит от подключения к базе данных

- ограниченный функционал

База данных (MySQL)

- больше возможностей
- хранение больших данных
- возможность сортировки

- нагрузка на сервер

Запись в файл на php

```
<?php
  /* Принимаем данные из формы */
  $sirname = $_POST["sirname"];
  $email = $_POST["email"];
  $text_message = $_POST["message"];
  // Открываем файл для записи
  $file = fopen ("base.txt","a+");
  if ( !$file )
  {
    echo("Ошибка открытия файла");
  }
  else
  {
    fputs ( $file, $sirname.“\r\n"); // Запись имени в файл
    fputs ( $file, $email.“\r\n"); // Запись email в файл
    fputs ( $file, $text_message.“\r\n"); // Запись текста сообщения в файл
    fputs ( $file, $text_message.“\r\n");
  }
  fclose ($file);
header("Location: ".$_SERVER["HTTP_REFERER"]); // Делаем редирект обратно
?>
```

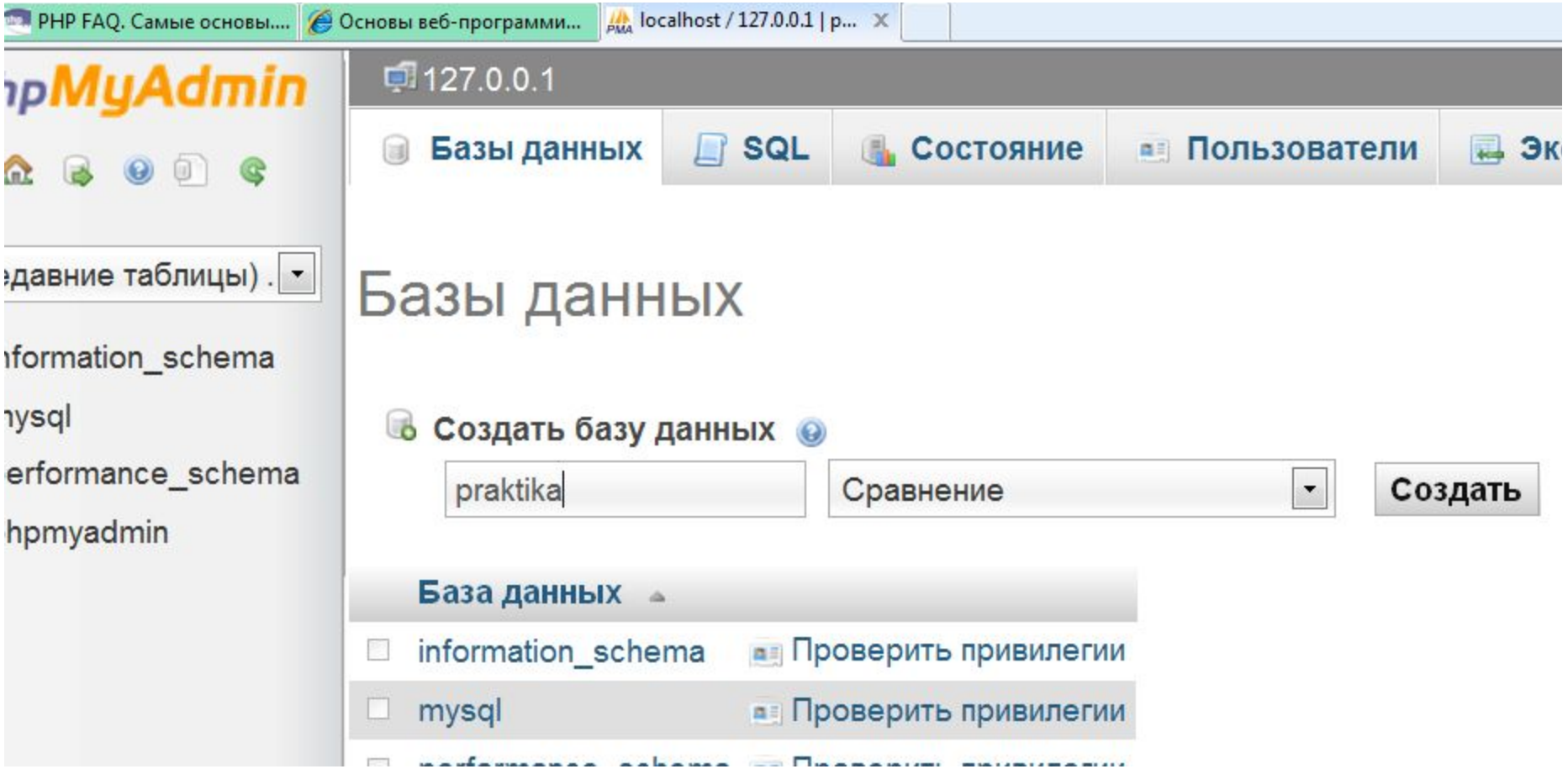
Чтение из файла на php

```
<?php
/* Отображаем комментарии на сайте */
$file = fopen("base.txt","r"); // Открываем файл для чтения
while ($i<100) {

if(!file)
    {
      echo("Ошибка открытия файла");
    }
    else
    {
      $buff = fgets ($file); // Чтение одной строки из файла base.txt
      print $buff;
      print "<br/>";
    }
    $i++;
}
fclose($file);
?>
```

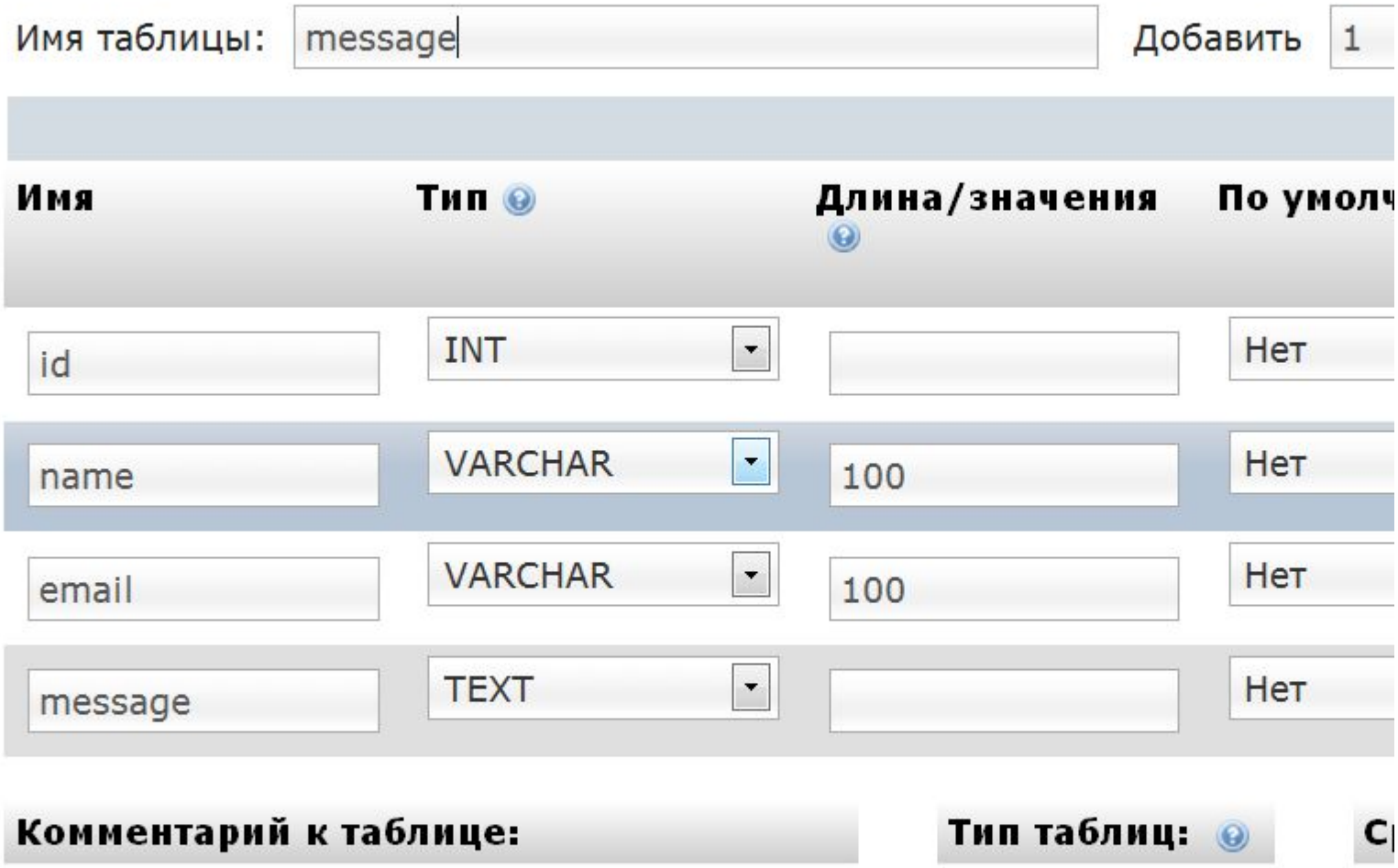
Использование БД Mysql

1. Создаём базу данных с использованием утилиты phpmyadmin (набираем в браузере localhost и выбираем phpmyadmin)



Использование БД Mysql

2. В созданной базе данных создаём таблицу (для этого нужно указать имя таблицы, количество полей(столбцов) и тип каждого поля).



Использование БД Mysql

3. Подключаемся к базе данных в php файле:

```
<?php
$db=mysql_connect("localhost", "root","");
mysql_select_db("praktika", $db);
?>
```

4. Записываем данные, введённые в форму

5. Отображаем (выбираем) данные из базы данных

Как записать данные в MySQL

Структурированный язык запросов (**S**tructured **Q**uery **L**anguage) – сокращённо SQL:

Есть четыре основных типа запросов данных в SQL:

SELECT – выбрать строки из таблиц;
INSERT – добавить строки в таблицу;
UPDATE – изменить строки в таблице;
DELETE – удалить строки в таблице;

Использование запроса SELECT для выборки нужных данных

SELECT column1, column2 FROM table_name;

Использование запроса INSERT для вставки новых данных

INSERT INTO table_name (column1, column2, column3) VALUES (‘data1’, ‘data2’, ‘data3’);

Как записать данные в MySQL

```
<?php
/* Принимаем данные из формы */
  $name = $_POST["sirname"];
  $email = $_POST["email"];
  $text_message = $_POST["message"];

/* Подключаемся к базе данных */
  $db=mysql_connect("localhost", "root","");
        mysql_select_db("praktika", $db);

/* Записывает данные */
$sql = "INSERT INTO message(name, email, message) VALUES
('$name', '$email', '$text_message')";
$result=mysql_query($sql) or die("Ошибка в запросе!".mysql_error());

/* Делаем редирект обратно */
header("Location: ".$_SERVER["HTTP_REFERER"]);
exit;
?>
```

Как выбрать данные из MySQL

```
<?php
/* Подключаемся к базе данных */
$db=mysql_connect("localhost", "root","");
mysql_select_db("praktika", $db);

/* Выбираем данные */
$sql="SELECT name, email, message FROM message";
$result=mysql_query($sql);

while ($line=mysql_fetch_row($result)) {
echo "<b>Имя:</b>".$line[0]."<br>";
echo "<b>Email:</b>".$line[1]."<br>";
echo "<b>Сообщение:</b>".$line[2]."<br>";
}
?>
```

Загрузка файлов на сервер

Приложение для загрузки файлов на сервер представляет собой HTML-форму (upload.html) и скрипт upload.php для ее обработки.

Загрузка файла на сервер осуществляется с помощью multipart-формы, в которой есть поле загрузки файла. В качестве параметра enctype указывается значение multipart/form-data:

```
<form action=upload.php method=post
enctype=multipart/form-data>
<input type=file name=uploadfile>
<input type=submit value=Загрузить></form>
```

Обработка multipart-форм

Как же PHP обрабатывает multipart-формы? Получив файл, он сохраняет его во временном каталоге upload_tmp_dir, имя файла выбирается случайным образом. Затем он создает четыре переменных суперглобального массива $_FILES. Этот массив содержит информацию о загруженном файле.

Содержимое массива $_FILES для нашего примера приведено ниже. Обратите внимание, что здесь предполагается использование имени uploadfile для поля выбора файла, в соответствии с приведенной выше multipart-форме. Разумеется, имя поля может быть любым.

- $_FILES['uploadfile']['name'] - имя файла до его отправки на сервер, например, pict.gif;
- $_FILES['uploadfile']['size'] - размер принятого файла в байтах;
- $_FILES['uploadfile']['type'] - MIME-тип принятого файла (если браузер смог его определить), например: image/gif, image/png, image/jpeg, text/html;
- $_FILES['uploadfile']['tmp_name'] (так мы назвали поле загрузки файла) - содержит имя файла во временном каталоге, например: /tmp/phpV3b3qY;
- $_FILES['uploadfile']['error'] - Код ошибки, которая может возникнуть при загрузке файла. Ключ *['error']* был добавлен в PHP 4.2.0.

Обработка multipart-форм

- После завершения работы скрипта, временный файл будет удален. Это означает, что мы должны его скопировать в другое место до завершения работы скрипта. То есть алгоритм работы сценария загрузки файла на сервер такой:
- Если кнопка "Submit" нажата, то файл уже будет загружен на сервер и его имя будут в переменной $_FILES['uploadfile']['name']. В этом случае скрипт должен сразу скопировать файл с именем $_FILES['uploadfile']['tmp_name'] в какой-нибудь каталог (необходимы права на запись в этот каталог).

Пишем upload.php

```
<?php

// Каталог, в который мы будем принимать файл:
$uploaddir = './files/';
$uploadfile = $uploaddir.basename($_FILES['uploadfile']['name']);

// Копируем файл из каталога для временного хранения файлов:
if (copy($_FILES['uploadfile']['tmp_name'], $uploadfile))
{
echo "<h3>Файл успешно загружен на сервер</h3>";
}
else { echo "<h3>Ошибка! Не удалось загрузить файл на сервер!</h3>"; exit; }

// Выводим информацию о загруженном файле:
echo "<h3>Информация о загруженном на сервер файле: </h3>";
echo "<p><b>Оригинальное имя загруженного файла: ".$_FILES['uploadfile']['name']."</b></p>";
echo "<p><b>Mime-тип загруженного файла: ".$_FILES['uploadfile']['type']."</b></p>";
echo "<p><b>Размер загруженного файла в байтах: ".$_FILES['uploadfile']['size']."</b></p>";
echo "<p><b>Временное имя файла: ".$_FILES['uploadfile']['tmp_name']."</b></p>";

?>
```

Считывание файлов из директории

```php
<?php
  $dir = 'images/'; // Папка с изображениями
  $files = scandir($dir); // Берём всё содержимое директории


  for ($i = 0; $i < count($files); $i++) { // Перебираем все файлы
    if (($files[$i] != ".") && ($files[$i] != "..")) { // Текущий каталог и родительский пропускаем

      $path = $dir.$files[$i]; // Получаем путь к картинке
        //Получив путь, мы можем выводить картинку в галлерею и слайдер, например,
        // <img src="<? echo $path ?>">
    }
  }
?>
```