Курсы
«Современные технологии программирования»

HTML

HyperText Markup Language
(Язык разметки гипертекста)

Торовец Н.Г.
natalia@kubsu.ru
2008 г.

Соглашение об авторских правах

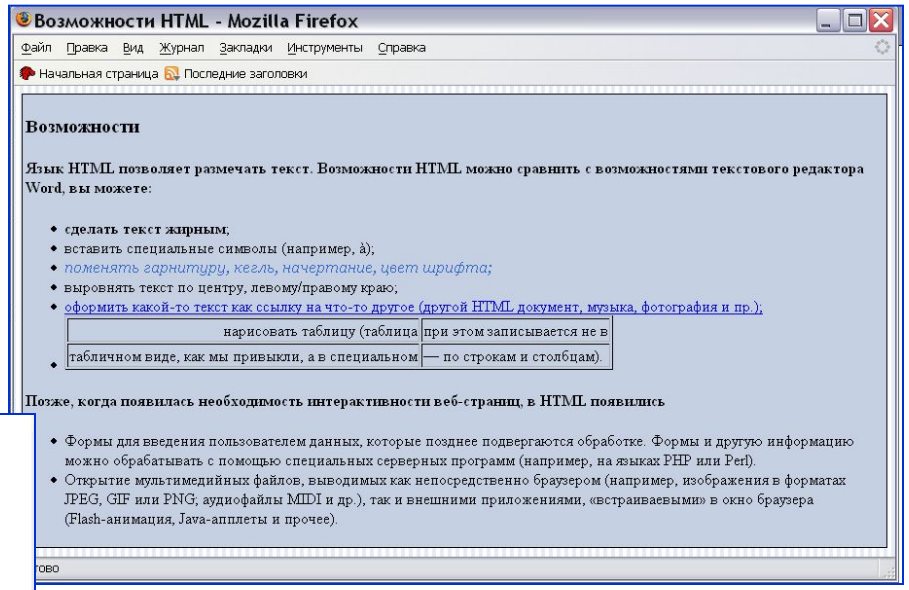
Этот материал предназначен исключительно для зарегистрированных в Интернет-центре КубГУ участников курсов, которые имеют право использовать его для самообучения, но не имеют права передавать его или его части другим лицам или использовать в коммерческих целях.

Воспроизведение материала лекции любым способом возможно только с письменного разрешения автора.

Что такое HTML?

HTML - *приложение SGML.*

Соответствует международному



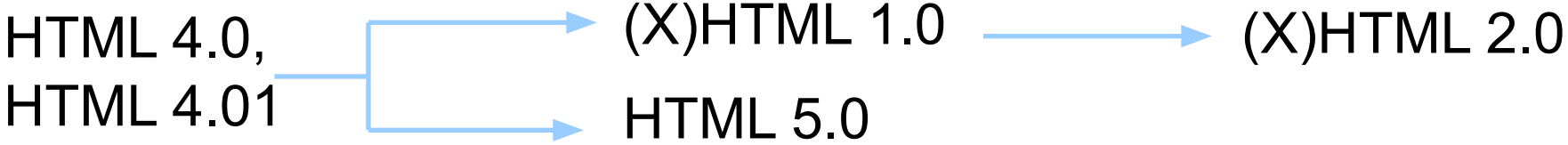

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//RU" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=win-1281" />
<title>Возможности HTML</title>
</head>

<body background="bgd.gif">
<div style="background: #C5D1E3; border:1px solid; padding:3px">
    <h3>Возможности</h3>
    <h4>Язык HTML позволяет размечать текст. Возможности HTML можно сравнить с возможностями текстового редактора Word, вы можете:</h4>
    <p>
        <ul>
            <li><b>сделать текст жирным</b>;
            <li>вставить специальные символы (например, &HxE0;);</li>
            <li><font color="#3366CC" face="Georgia, Times New Roman, Times, serif" ><i>поменять гарнитуру, кегль, начертание, цвет шрифта;</i></font></li>
            <li>выровнять текст по центру, левому/правому краю; </li>
            <li><a href="#">оформить какой-то текст как ссылку на что-то другое (другой HTML документ, музыка, фотография и пр.);</a></li>
            <li>
                <table border="1px solid">
                    <tr>
                        <td align="right">нарисовать таблицу (таблица </td>
                        <td>при этом записывается не в </td>
                    </tr>
                    <tr>
                        <td>табличном виде, как мы привыкли, а в специальном</td>
                        <td> — по строкам и столбцам).</td>
                    </tr>
                </table>
            </li>
        </ul>
    </p>
    <h4>Позже, когда появилась необходимость интерактивности веб-страниц, в HTML появились</h4>
    <p>
        <ul>
            <li>Формы для введения пользователем данных, которые позднее подвергаются обработке. Формы и другую информацию можно обрабатывать с помощью специальных
серверных программ (например, на языках PHP или Perl).</li>
            <li>Открытие мультимедийных файлов, выводимых как непосредственно браузером (например, изображения в форматах JPEG, GIF или PNG; аудиофайлы MIDI и др.), так
 и внешними приложениями, «встраиваевыми» в окно браузера (Flash-анимация, Java-апплеты и прочее).</li>
        </ul>
    </p>
</div>
</body>
</html>
```

HTML-документ - ASCII-файл, доступный для просмотра и редактирования в любом текстовом редакторе. Отличием от обычного текстового файла является наличие в HTML-документах специальных команд - **тэгов**, которые указывают правила форматирования документа

История и стандарты HTML

Язык HTML был разработан британским учёным Тимом Бернерсом-Ли приблизительно в 1991-1992 годах.



Конструкции, используемы в HTML

- Элементы; [<el_name>][contents][</el_name>]

<P>нечувствителен</P>
к регистру
<DIV><P>без закрывающего тега </DIV>

ОТД HTML указывает для каждого типа элементов, требуются ли начальный и конечный теги.

Элементы - это не теги.

- Атрибуты;

<H1 id="h1"> с атрибутом </H1>

Значения атрибутов должны быть ограничены с использованием “” или ‘’.

- Ссылки-мнемоники;

"&#? " – Специалный символ
">" - знак >
""" - знак "

где ? - это код ISO 8859-1

- Комментарии.

Комментарии - это разметка.

<!– это комментарий -->
// и это комментарий
/* это тоже комментарий */

Структура документа

```
<!DOCTYPE HTML PUBLIC
"-//W3C//DTD HTML
4.01//EN"
"http://www.w3.org/TR/html4/
strict.dtd"> // пролог HTML
<HTML> // тип документа
  <HEAD>
    // голова документа
    <TITLE>Заголовок
документа</TITLE>
  </HEAD>
  <BODY>
      // тело документа
  </BODY>
</HTML>
```

Документ в формате HTML 4.0 состоит из трех частей:

- строки, содержащей информацию о версии документа,
- раздела заголовков (определяемого элементом HEAD),
- тела (может вводиться элементом BODY или FRAMESET), которое включает содержимое документа.

Информация о версии HTML

Объявление типа документа указывает определение типа документа (DTD), используемое в этом документе:

- **HTML 4.0 Strict DTD** (строгое определение) - все элементы и атрибуты, не являющиеся нежелательными и не использующиеся в документах с кадрами. : <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN" "http://www.w3.org/TR/REC-html40/strict.dtd">
- **HTML 4.0 Transitional DTD** (переходное определение) - все, что включено в строгое DTD, а также нежелательные элементы и атрибуты : <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" "http://www.w3.org/TR/REC-html40/loose.dtd">
- **HTML 4.0 Frameset DTD** (определение для кадров) - все, что включено в переходное DTD, а также кадры. : <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN" "http://www.w3.org/TR/REC-html40/frameset.dtd">

Раздел заголовков

Элемент HEAD содержит информацию о текущем документе, такую как заголовок (TITLE), ключевые слова (META), которые могут использоваться поисковыми машинами, и другие данные, которые не считаются содержимым документа.

Каждый документ HTML **должен** иметь элемент TITLE в разделеHEAD.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN" "http://www.w3.org/TR/REC-html40/strict.dtd">
<HTML> <HEAD> <TITLE>Динамика популяции</TITLE> <LINK rel="stylesheet" type="text/css" href="smartstyle.css"> </HEAD>
<BODY> ... тело документа... </BODY> </HTML>
```

Метаданные. Тег META

Каждый элемент META задает пару свойство/значение.

```
<META name="Author"
lang= "ru" content="Arnaud
Le Hors">
//информация об авторе
и языке
<META
http-equiv="Content-Type"
content="text/html;
charset=win-1251">
//указание информации
по умолчанию
```

Атрибуты META:

- name – имя свойства;
- content – значение свойства;
- scheme - имя схеы;
- http-equiv – (может применяться вместо name) используется серверами HTTP для сбора информации для заголовков сообщений ответов HTTP;
- lang – язык;
- dir – направление текста.

Метаданные. Примеры

```
<META HTTP-EQUIV="AUTHOR" CONTENT="My Name">
//информация об авторе
<META HTTP-EQUIV="REPLY-TO"
CONTENT="J.master@narod.ru"> //обратная связь
<META HTTP-EQUIV="DESCRIPTION" CONTENT="My
Description"> //описание своего документа
<META HTTP-EQUIV="KEYWORDS"
CONTENT="My,Html,Home,MyHtml"> // ключевые слова для
роботов-поисковиков
<META NAME="DATA" CONTENT="date_of_create">
//дата создания документа
<META HTTP-EQUIV="PRAGMA" CONTENT="NO-CACHE">
//не хранить в кэше
```

Тело документа. Атрибуты

Нежелательные:

- background - фоновое изображение.
- text - цвет текста.
- link - цвет текста не посещённых гипертекстовых ссылок.
- vlink - цвет текста посещённых ссылок.
- alink - цвет текста активной (текущей) ссылки.

Определяемые в любом другом месте:

- id, class - идентификаторы в пределах документа;
- lang - информация о языке;
- dir - направление текста;
- title - заголовок элементаж;
- style - встроенная информация о стиле;
- bgcolor - цвет фона.

Тело документа (1/2)

Использовать нежелательно

```
…
<HEAD>
<TITLE>Динамика популяции</TITLE>
 <BODY
bgcolor="white"
text="black" link="red"
alink="fuchsia"
vlink="maroon">
 ... тело документа...
 </BODY>
…
```

Желательно использовать

```
…
<HEAD>
 <TITLE>Динамика популяции</TITLE>
 <STYLE type="text/css">
 BODY { background: white; color: black}
 A:link { color: red }
 A:visited { color: maroon }
 A:active { color: fuchsia }
 </STYLE>
 </HEAD>
 <BODY> ... тело документа... </BODY>
 …
```

Способы задания пути

1. background="http://www.arhiv.ru/fp_0008.gif" - рисунок с другого сайта;
2. background="C:/users/arhiv.www/fp_0008.gif" - рисунок на вашем компьютере;
3. background="pic/fp_0008.gif" - рисунок из директории pic;
4. background="../fp_0008.gif" - рисунок из родительской директории;
5. background="fp_0008.gif" - рисунок из той же директории;

1,2 - абсолютный путь (жестко задаются, привязаны к определенному сайту или компьютеру)

3,4,5 - относительный путь (рекомендуются, хорошо переносимы на другие компьютеры)

При размещении документа в интернете регистр букв имени имеет значение.

Цветовые шкалы

RGB		Слово-синоним
#000000	Rgb(0,0,0)	Black
#ff0000	Rgb(255,0,0)	Red

- Вместо элементов и атрибутов HTML для указания цвета желательно использовать таблицы стилей.
- Не следует использовать комбинации цветов, вызывающие проблемы у пользователей.
- При использовании изображения в качестве фона или установлении цвета фона, нужно установить и цвета текста.
- Цвета, указанные в элементах BODYи FONT и в bgcolor в таблицах выгладят по-разному на разных платформах.
- При возможности нужно принимать общие соглашения.

Фразовые элементы (1/4)

- **BLOCKQUOTE** – предназначен для длинных цитат, выделяются отступами (содержимое уровня блока).
- **Q** предназначен для коротких цитат (встроенное содержимое), в которых не нужно разбиение на абзацы, выделяется кавычками.
- **SUB** – нижний индекс.
- **SUP** – верхний индекс.
- **Заголовки** - H1 (сам

```
H<sub>2</sub>O E = mc<sup>2</sup>
<SPAN lang="fr">M<sup>lle</sup>
Dupont</SPAN>
```


 – переход на новую строку

```
<BLOCKQUOTE
cite="http://www.mycom.com/tolkien/twotowers.html"> <P>They went in single file, running like hounds on a strong scent, and an eager light was in their eyes.</P> </BLOCKQUOTE>
```

Фразовые элементы (2/4)

- **EM** - Выделение.
- **STRONG** - Более сильное выделение.
- **CITE** - Цитата или ссылка на другие ресурсы.
- **DFN** - Определение вложенного термина.
- **CODE** - Компьютерный код.
- **SAMP** - Пример программ, сценариев и т.д.
- **KBD** - Текст, который должен ввести пользователь.
- **VAR** - Экземпляр переменной или аргумента программы.
- **ABBR** - Сокращенная форма .
- **ACRONYM** - Акроним (например, WAC, радар и т.д.).

Атрибуты:

id, class, lang, title, style

Фразовые элементы (3/4)

- Элемент **PRE** сообщает браузеру, что содержащийся в нем текст "отформатирован".

```
<PRE> Higher still and higher From the earth thou springest Like a cloud of fire; The blue deep thou wingest, And singing still dost soar, and soaring ever singest.
</PRE>
```

- ADDRESS может использоваться для указания контактной информации или основной части документа, такой как форма

```
<ADDRESS>
 <P>
<A href="..">Dave Raggett</A>, <A href="..">Arnaud Hors</A>, contact persons for the <A href="Activity">W3C HTML Activity</A><BR>
</ADDRESS>
```

Фразовые элементы (4/4)

```
<P> <ABBR title="World Wide Web">WWW</ABBR>
<ABBR lang="fr" title="Soci&eacute;t&eacute; Nationale des
Chemins de Fer"> SNCF </ABBR> <ABBR lang="es"
title="Do&ntilde;a">Do&ntilde;a</ABBR> <ABBR
title="Abbreviation">abbr.</ABBR>
```

WWW SNCF Doña abbr.

```
Как сказал <CITE>Гари Трумэн</CITE>, <Q
lang="en-US">The buck stops here.</Q> Подробнее см.
<CITE>[ISO-0000]</CITE>. <br /> В дальнейшем
используйте следующий номер ссылки:
<STRONG>1-234-55</STRONG>
```

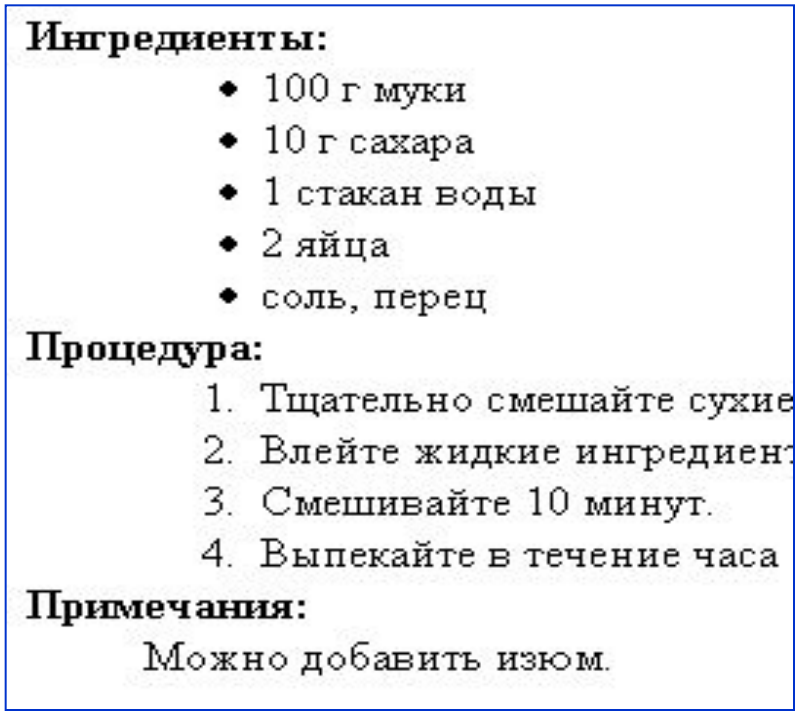
Как сказал *Гари Трумэн*, “The buck stops here.” Подробнее см. *[ISO-0000]*.
В дальнейшем используйте следующий номер ссылки: **1-234-55**

Списки

- <UL> - Неупорядоченные списки (элемент – <LI>),
- <OL> - упорядоченные списки (элемент – <LI>),
- списки определений <DL> (термин – <DT>, определение – <DD>)

Атрибуты:

- type, start, value;
- id, class, lang, dir, title, style.



Ингредиенты:

- 100 г муки
- 10 г сахара
- 1 стакан воды
- 2 яйца
- соль, перец

Процедура:

1. Тщательно смешайте сухие
2. Влейте жидкие ингредиен
3. Смешивайте 10 минут.
4. Выпекайте в течение часа

Примечания:

Можно добавить изюм.

```
<DL>
 <DT>Ингредиенты:
  <DD>
   <UL>
    <LI>100 г муки </LI>…
  <DT>Процедура:
  <DD>
   <OL>
   <LI>Смешайте ингредиенты.
</LI>…
  <DT>Примечания:
  <DD>Можно добавить изюм.
 </DL>
```

Ссылки. Элемент A

Каждый элемент A определяет **якорь**:

- Содержимое элемента A определяет положение якоря.
- Атрибуты name и id задают имя якоря, так что он может служить пунктом назначения любого числа ссылок.
- Атрибут href назначает якорь пунктом назначения ровно одной ссылки.

Атрибуты:

- name ,href, hreflang, type,charset, id, class, lang, dir, title, style, shape и coords (навигационные карты), target (информация о целевом кадре), rel и rev (прямая и обратная ссылки), tabindex (последовательность перехода), accesskey (клавиши доступа).

Подробнее о W3C Вы можете узнать на <A href="http://www.w3.org/"> Web-сайте W3C</A>.

Подробнее о W3C Вы можете узнать на Web-сайте W3C.

Ссылки. Элемент Link

Link определяет связь. Может присутствовать только в разделе HEAD документа (неограниченное число раз). Хотя элемент LINK, но содержит информацию об отношениях.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html40/strict.dtd">
<HTML>
 <HEAD>
 <TITLE>Глава 2</TITLE>
 <LINK rel="Index" href="../index.html">
 <LINK rel="Next" href="Chapter3.html">
 <LINK rel="Prev" href="Chapter1.html">
 </HEAD> ...продолжение документа...
```

LINK может, в частности, использоваться для указания информации для поисковых машин и для ссылок на внешние таблицы стилей.

Объекты

Для добавления разных объектов можно использовать элемент OBJECT

Атрибуты:

- id, class, lang, dir, title, style, tabindex, usemap (клиентские навигационные карты) name (предоставление формы) , align, width, height, border, hspace, vspace (визуальное представление объектов, изображений и апплетов).

```
<BODY>
<P>Около Большого Каньона:
<OBJECT data="canyon.png" type="image/png">
<EM>Около</EM> Большого Каньона.
</OBJECT>
</BODY>
```

Является обобщением Для APPLET, IMG и IFRAME.

```
<P><OBJECT classid="http://www.miamachina.it/analogclock.py">
Часы с анимацией.
</OBJECT>
```

Изображения

Элемент IMG позволяет включить изображение.

Атрибуты:

- src - задает местоположение изображения, longdesc - определяет ссылку на длинное описание изображения id, class, alt, lang, dir, title, style, ismap, usemap, name (предоставление формы) , align, width, height, border, hspace, vspace.

```
<BODY>
<P>Я только что вернулся из отпуска! Вот фотография моей семьи на озере:
<IMG src="http://www.somecompany.com/People/Ian/vacation/family.png"
     alt="Фотография моей семьи на озере.">
</BODY>
```

```
<BODY>
<P>Я только что вернулся из отпуска! Вот фотография моей семьи на озере:
<OBJECT data="http://www.somecompany.com/People/Ian/vacation/family.png"
        type="image/png">
Фотография моей семьи на озере.
</OBJECT>
</BODY>
```

Атрибут style

С помощью атрибута style можно задать стилевое отображение элементов страницы HTML.

Некоторые атрибуты:

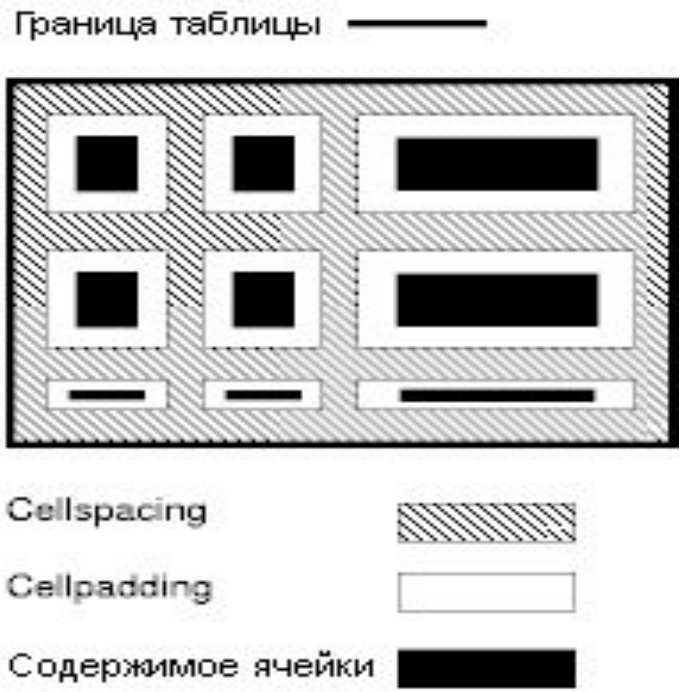
- width - ширина;
- height - высота;
- border - граница;
- background - фон.

```
<div style="width:300px; height:40px; border:1px
solid #333399; background: #CCFFFF;">
</div>
```

Подробнее стилях будем говорить позже.

Таблицы

- Таблица: TABLE
- Заголовки таблицы: THEAD,
- Нижние заголовки: TFOOT
- Раздел таблицы: TBODY
- Строка таблицы: TR
- Ячейка таблицы: TD
- Заголовок таблицы: TH
- Подпись: CAPTION



Атрибуты:
border, align, valign, nowrap, colspan (объединение столбцов), rowspan (объединение строк), summary, id, class, lang, dir, title, style, bgcolor, frame, rules, border, cellspacing, cellpadding и др.

Таблицы (1/2)

```
<TABLE border="1"
          summary="В этой таблице приводится не
                                средняя высота и ве
<CAPTION><EM>Тестовая таблица с объединенными я
<TR><TH rowspan="2"><TH colspan="2">Средний
    <TH rowspan="2">Красные<BR>глаза
<TR><TH>высота<TH>вес
<TR><TH>мужской пол<TD>1.9<TD>0.003<TD>40%
<TR><TH>женский пол<TD>1.7<TD>0.002<TD>43%
</TABLE>
```

Тестовая таблица с объединенными ячейками

	Средний		Карие глаза
	рост	вес	
Мужчины	1.9	0.003	40%
Женщины	1.7	0.002	43%

Табличная вёрстка

Табличная вёрстка – представление структуры страницы в виде сетки с использованием таблиц.


```
<BODY>
 <TABLE id="table">
  <TR>
   <TD id="d1">
       Меню 1
   </TD>
  </TR>
 <TR>
   <TD id="d2">
       Меню 2
   </TD>
  </TR>
 <TR>
   <TD id="d3">
       Меню 3
   </TD>
  </TR>
 </TABLE>
</BODY>
```

Преимущества:

- Структурированность.

Недостатки:

- Большое количество кода;
- Трудная правка
- Негибкий дизайн в разработке
- Такие сайты хуже индексируются поисковыми роботами.
- Плохая совместимость с некоторыми устройствами.

Слои

Слой - это HTML-контейнер (тег DIV или SPAN), в который можно помещать желаемое содержимое для последующего точного позиционирования на странице.

SPAN определяет встраиваемую информацию

DIV определяет информацию уровня блока

```
<div>Слой 1
<span>Слой 2</span>
</div>
```

Атрибуты:

- id, class (идентификаторы в пределах документа);
- lang (информация о языке);
- dir (направление текста);
- title (заголовок элемента);
- style (встроенная информация о стиле);
- align (выравнивание).

Блочная вёрстка

Блочная вёрстка – представление страницы в виде блоков - слоёв (div) html, а для задания визуального форматирования – стилей CSS.


```
<BODY>
    <DIV id="d1">
        Меню 1
    </DIV>
    <DIV id="d2">
        Меню 2
    </DIV>
    <DIV id="d3">
        Меню 3
    </DIV>
</BODY>

</html>
```

Преимущества:

- Значительное упрощение построения страниц;
- Упрощение добавления, исправления содержимого и изменения внешнего вида.

Недостатки:

- Плохая переносимость подобных страниц браузерами устаревшей конструкции.
- Некоторые недостатки связаны с CSS

Формы (1/5)

Форма HTML - это раздел документа, в котором содержатся обычная информация, разметка и специальные элементы, называемые управляющими элементами(флажки, кнопки с зависимой фиксацией, меню и т.д.), а также метки этих управляющих элементов.

Формы (2/6)

Тэг	Атрибут	Назначение
<FORM>	... ACTION="URL" METHOD=GET | POST	по событию SUBMIT передать на URL по методу GET | POST
	... ENCTYPE="multipart/form-data" ...>	многостраничная
<INPUT	.. NAME="_name_"...	имя поля
	... VALUE="_*value*_“	значение поля по умолчанию
	... TYPE="TEXT" |	
	... TYPE="PASSWORD" |	поле ввода пароля
	... TYPE="CHECKBOX" |	
	... TYPE="RADIO" |	
	... TYPE="IMAGE" |	Изображение

Формы (3/6)

Тэг	Атрибут	Назначение
<INPUT...	... TYPE="HIDDEN" |	Скрытое поле
	... TYPE="SUBMIT" |	кнопка передачи данных формы на сервер
	... TYPE="RESET" |	кнопка "очистки" формы
	... TYPE="BUTTON" ...	просто кнопка
	... TYPE="FILE" ...	поле для файла
	... CHECKED="CHECKED" ...	"включено" по умолчанию - для checkboxes и radio boxes
	... SIZE=?? ...	размер поля в символах для текстовых полей
	... MAXLENGTH=?? ...	максимальное количество символов в полях ввода
... >		

Формы (4/6)

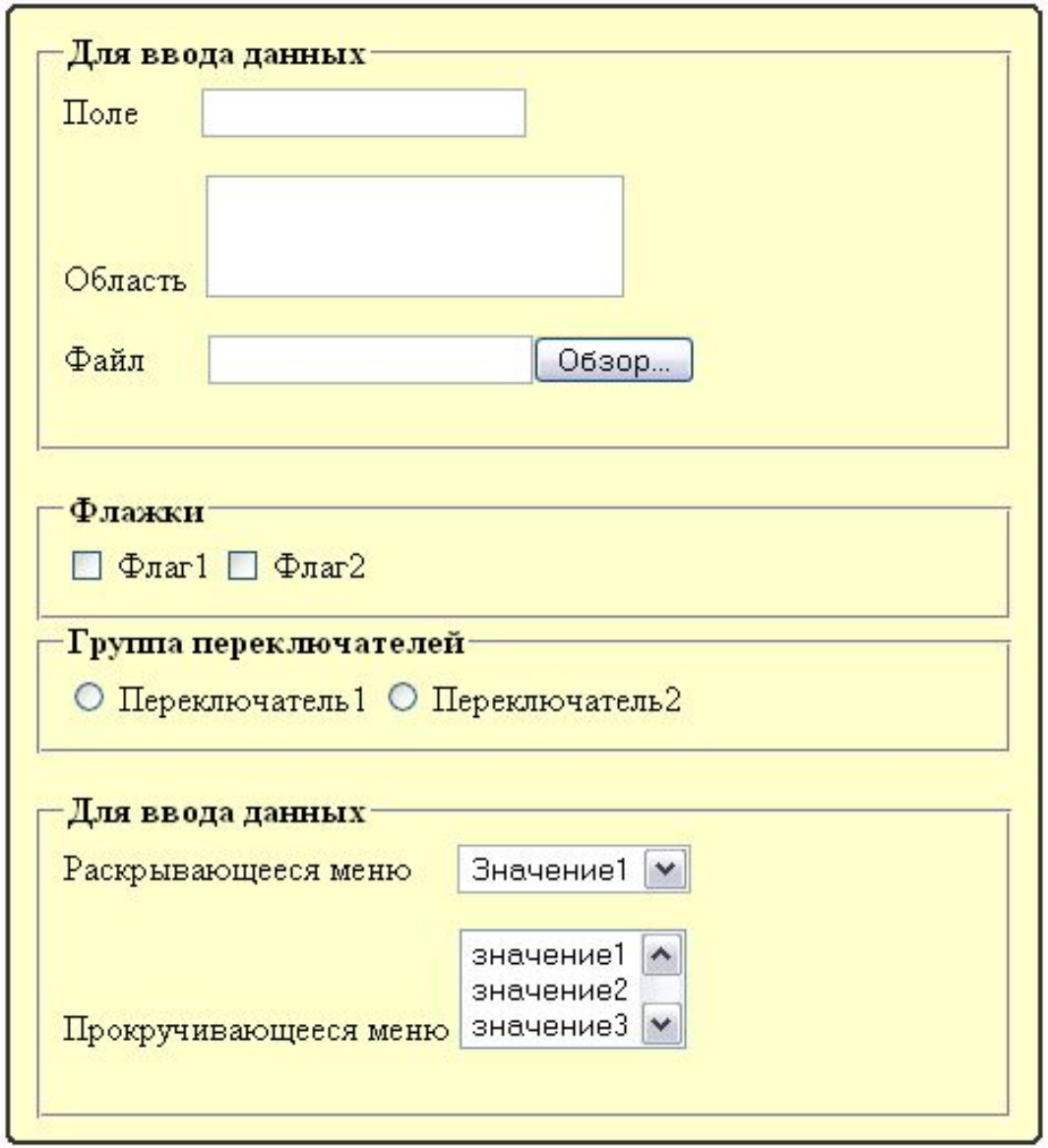
Тэг	Атрибут	Назначение
<SELECT>		Список вариантов
	... NAME="***“…	Имя списка
	… SIZE=“?”…	Число вариантов
	…MULTIPLE=“MULTIPLE”…	выбор нескольких элементов
<OPTION>		элемент списка выбора
	… SELECTED=“SELECTED”…	Опция по умолчанию
</OPTION>		
</SELECT>		

Формы (5/6)

Тэг	Атрибут	Назначение
<TEXTAREA	…ROWS=“?” COLS=“?”…	Ввод текста в блок размерами ?? строк x ?? колонок
	… NAME="***“…	Имя текста
	… WRAP=“OFF | VIRTUAL | PHYSICAL”…	Разбивка на строки
></TEXTAREA>		
<FIELDSET>		Группирует элементы формы
<LEGEND> </LEGEND>		Заголовок группы
</FIELDSET>		
</FORM>		

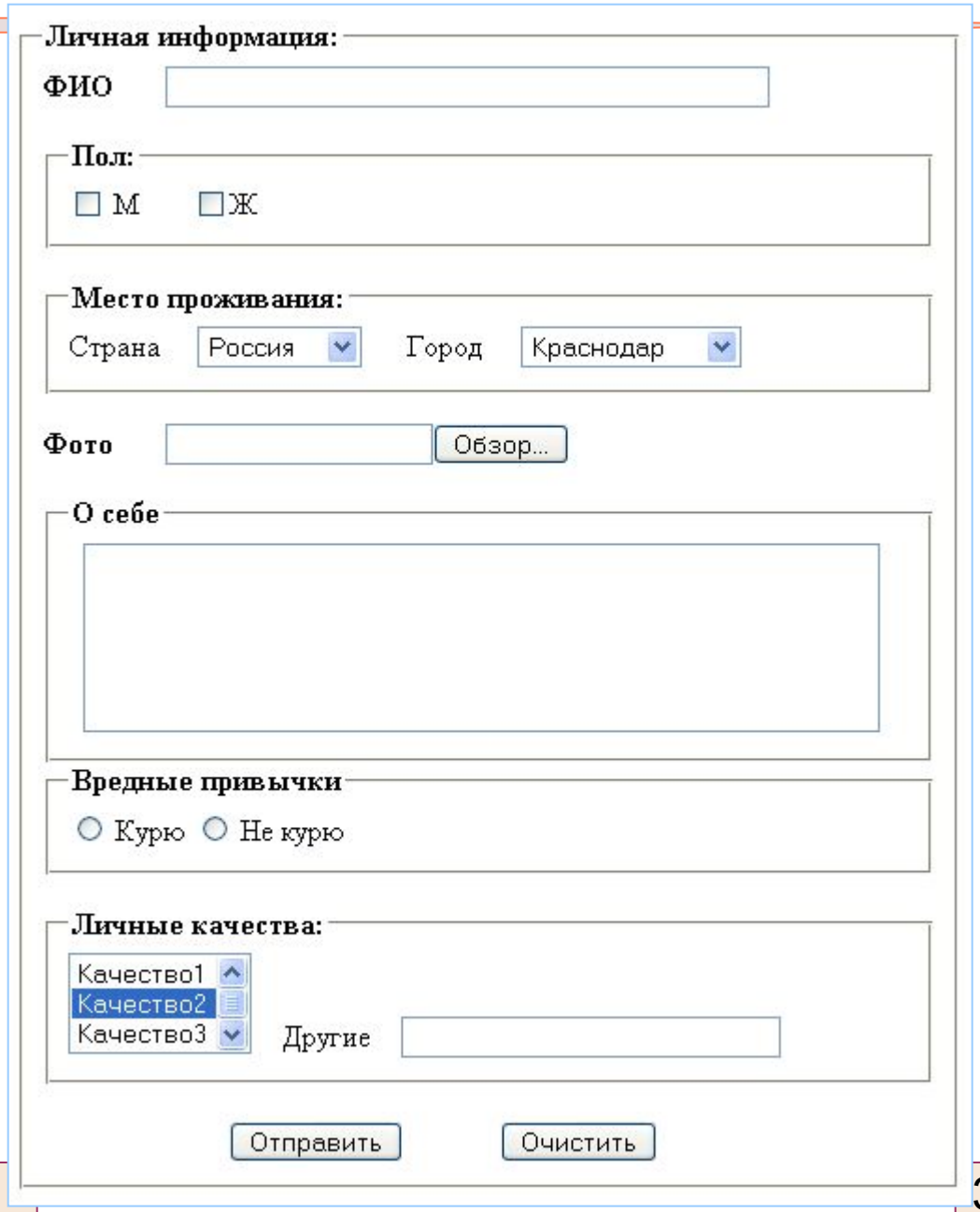
Формы (6/6)

```
<fieldset><legend><strong>Для
данных</strong></legend>…
<input type="text" name="textfie
<textarea name="textarea"></text
<input type="file" name="file" />.
</fieldset>…
<input type="checkbox"
name="checkbox" value="checkbox" /
<input type="checkbox"
name="checkbox2" value="checkbox"
<input name="Radio_Group"
type="radio" value="radiobutton"
/>Переключатель1</label>…
<select name="select" size="1">.
<option>Значение3</option>
</select></label></p>
<p>Прокручивающееся меню
<select name="menu1" size='
<option>значение4</option>.
```



Задание

Создать форму следующего вида:



Карты изображений (Image Map)

Чтобы включить поддержку карты для изображения, необходимо ввести указать параметр:

USEMAP="*url#map_name*"

Атрибуты:

- shape (= default|rect|circle|poly) - определяет форму области,
- coords - определяет положение формы на экране (число и порядок значений зависят от определенной формы),
- nohref - указывает, что с этой областью ссылка не связана,
- usemap - связывает навигационную карту с элементом. Навигационная карта определяется с помощью элемента MAP. Значение атрибута usemap должно совпадать со значением атрибута name связанного элемента MAP.
- name, id, class, lang, dir, title, style, name, ale, href, tabindex, accesskey.

Карты изображений (1/2)

```
<P><IMG src="navbar1.gif" usemap="#map1" alt="навигационная панель">
<MAP name="map1">
 <AREA href="guide.html"
          alt="Руководство по доступу"
          shape="rect"
          coords="0,0,118,28">
 <AREA href="top10.html"
          alt="Первые десять"
          shape="poly"
          coords="276,0,373,28,50,50,276,0">
</MAP>
```

```
<P><OBJECT data="navbar1.gif" type="image/gif" usemap="#map1">
</OBJECT>

...продолжение страницы...

<MAP name="map1">
  <P>Перемещение по узлу:
  <A href="guide.html" shape="rect" coords="0,0,118,28">Руководство по доступу</a> |
  <A href="shortcut.html" shape="rect" coords="118,0,184,28">Переход</A> |
  <A href="search.html" shape="circle" coords="184,200,60">Поиск</A> |
  <A href="top10.html" shape="poly" coords="276,0,373,28,50,50,276,0">Первые десять</A>
</MAP>
```

Преимущества и недостатки HTML

Преимущества:

- Межплатформенность;
- Малый информационный объём.

Недостатки:

- Отсутствие возможности форматирования (стилевого оформления).

Перспективы

В HTML 5 представлен ряд новых тегов:

- формально подобных "div" и "span", но отличающихся семантически ("nav", "header" и "footer") – логическое разделение контента, помощь в индексировании документа поисковыми системами и изменения представления на устройствах с небольшими экранами
- существенно расширены мультимедиа возможности (тэги "audio" и "video ").

Отличия от HTML 4:

- Новые правила парсинга;
- Новые элементы: section, video, progress, nav, meter, time, aside, canvas;
- Новые атрибуты полей ввода (Input): time, email, url;
- Новые атрибуты: ping, charset, async;
- Глобальные атрибуты применимые для всех элементов документа: id, tabindex, repeat; *
- Прекращена поддержка элементов: center, font, strike.

XHTML

XHTML - EXtensible HyperText Markup Language - (Расширенный язык разметки гипертекста).

Правила XHTML следующие:

- Все теги и параметры должны быть набраны в нижнем регистре (строчными символами).
- Значения любых параметров необходимо заключать в кавычки.
- Требуется закрывать все теги, даже такие, которым не сопоставлен закрывающий тег.
- Должна соблюдаться правильная вложенность тегов.
- Нельзя использовать сокращенные атрибуты тегов.
- Вместо параметра name необходимо указывать атрибут id.
- Следует определять DTD (document type definition, описание типа документа) с помощью тега <!DOCTYPE>.

Микроформаты

Позволяют включать дополнительную информацию для поисковых роботов. Существующие стандарты (X)HTML позволяют включать семантические пометки при помощи следующих HTML атрибутов: class, rel, rev

```
<div>
    <div>Василий Пупкин</div>
    <div>Рога и Копыта</div>
    <div>495-564-1234</div>
    <a href="http://vasya.ru/">http://vasya.ru/</a>
</div>
```

С разметкой hCard:

```
<div class="vcard">
    <div class="fn">Василий Пупкин</div>
    <div class="org">Рога и Копыта</div>
    <div class="tel">495-564-1234</div>
    <a class="url" href="http://vasya.ru/">http://vasya.ru/</a>
</div>
```

Микроформаты (1/2)

Существующие микроформаты:

- hCalendar - для событий
- hCard - для контактной информации, включая:
- adr - для почтовых адресов
- geo - для географических координат (широта;долгота)
- hReview - для обзоров
- hResume - для резюме
- rel-directory - для распределенного создания каталогов
- rel-tag - для децентрализованных пометок (тэгов), см. также фолксономия.
- xFolk - для помеченных ссылок
- XFN - для социальный взаимоотношений
- XOXO - для списков,
- microformats.org – микроформаты.

CSS

Cascading Style Sheets

(Каскадные таблицы стилей)

Назначение

CSS предназначен для разделения логической структуры документа и формы его представления. Логическая структура документа определяется элементами HTML-разметки, а форма представления каждого из этих элементов задается CSS-описателем элемента.
CSS позволяет полностью переопределить форму представления элемента разметки по умолчанию.

```
<i>Отобразим текст курсивом</i>
```

Отобразим текст курсивом

```
<i style="text-decoration:underline;font-style:normal;">
Отобразим текст курсивом
</i>
```

<u>Отобразим текст курсивом</u>

Элементы и атрибуты языка документа

- В CSS имена свойств, дескрипторов и псевдоклассов с двух сторон ограничиваются одинарными кавычками.
- В CSS значения с двух сторон ограничиваются одинарными кавычками.
- Имена элементов языка документа пишутся прописными буквами.
- Имена атрибутов языка документа пишутся строчными буквами и с двух сторон ограничиваются двойными кавычками.

Способы применения CSS

- переопределение стиля в элементе разметки
- размещение описания стиля в заголовке документа в элементе STYLE
- размещение ссылки на внешнее описание через элемент LINK
- импорт описания стиля в документ

Переопределение стиля

Применение атрибута STYLE у данного элемента разметки

```
<h1 style="font-weight:normal;
font-style:italic;
font-size:10pt;">
Заголовок первого уровня
</h1>
```

Заголовок первого уровня

Атрибут style можно применить внутри любого элемента разметки.

Но не всякие параметры стиля можно установить для конкретного элемента разметки.

Элемент STYLE

Элемент STYLE позволяет определить стиль отображения для:

- стандартных элементов HTML-разметки
- произвольных классов (селектор class)
- HTML-объектов (селектор id)

```
<head>
 <style> стандартные элементы разметки описываются
p { color:darkred; text-align:justify; font-size:8pt; }
</style> в элементе STYLE
</head>
<body>
```

Ссылка на внешнее описание

Осуществляется при помощи элемента LINK, который размещают в элементе HEAD.

```
<link type="text/css"
rel="stylesheet"
href="http://kuku.ru/my_css.css">
```

Rel обязан иметь значение "stylesheet".
Type может принимать значения: "text/css" или "text/javascript“.
Атрибут href задает универсальный локатор ресурса (URL) для внешнего файла описания стилей. Это может быть ссылка на файл с любым именем, а не только на файл с расширением *.css

Импорт описания стилей

Импортировать стиль можно либо внутрь элемента STYLE, либо внутрь внешнего файла, который представляет собой описатель стиля. Оператор импорта стиля должен предшествовать всем прочим описателям

```
<style>
@import:url(http://kuku.ru/style.css)
a { color:cyan; text-decoration:underline; }
</style>
```

Импортируемый стиль можно переопределить либо через описатель элемента в STYLE, либо через атрибут элемента style

Типы носителей

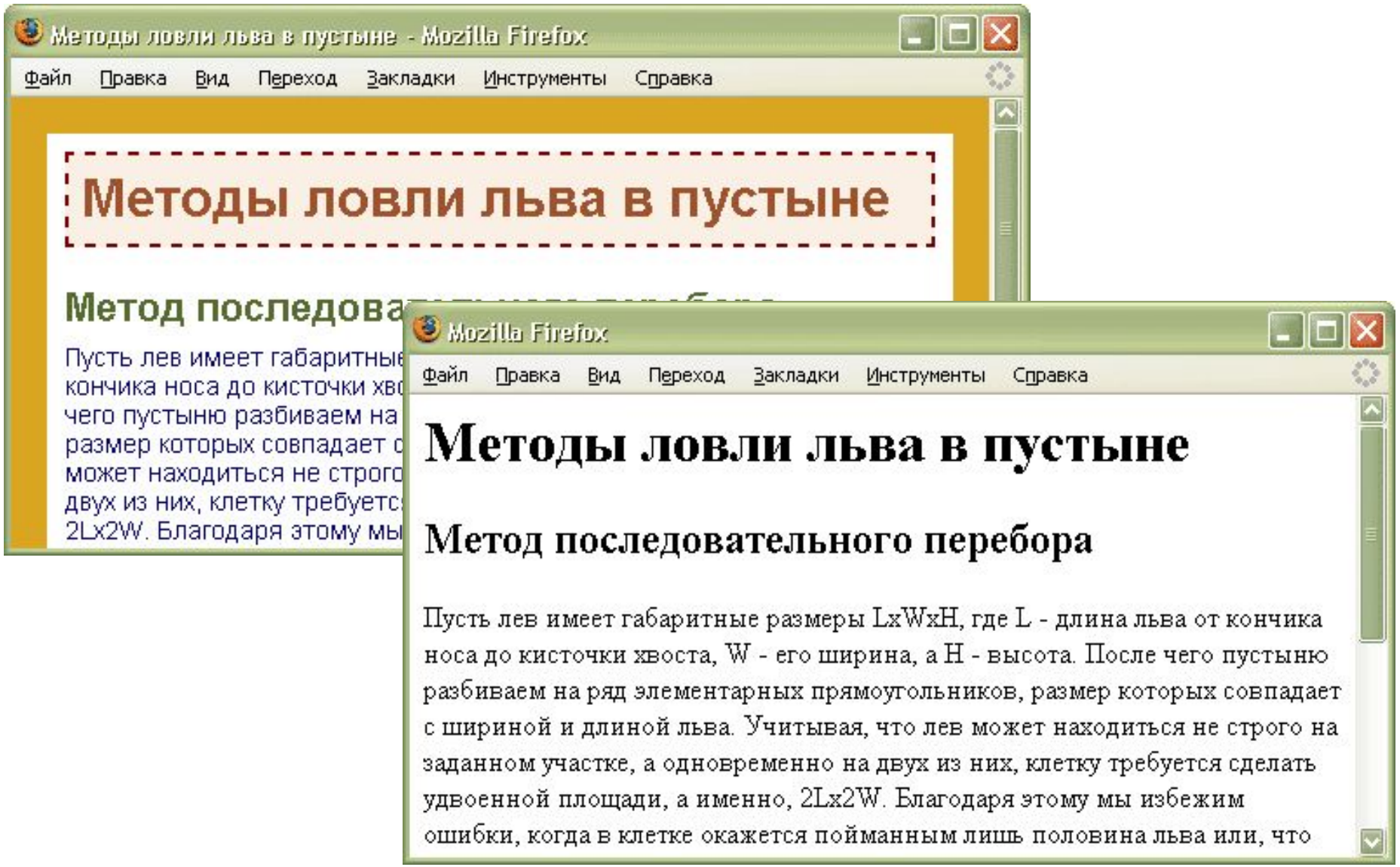
- all - Все типы. Это значение используется по умолчанию.
- a текста вслух. Сюда, например, можно отнести речевые браузеры.
- braille Устройства, основанные на системе Брайля, которые предназначены для слепых людей.
- handheld - Наладонные компьютеры и аналогичные им аппараты.
- print - Печатающие устройства вроде принтера.
- projection - Проектор.
- screen - Экран монитора.
- tv - Телевизор.

Типы носителей (1/2)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<title>Импорт стиля</title>
<style type="text/css">
 @import "/style/main.css" screen; /* Стиль для вывода результата на монитор */
 @import "/style/palm.css" handheld, print; /* Стиль для печати */
</style>
</head>
<body>
```

```
<style type="text/css">
@media screen { /* Стиль для отображения в браузере */
 BODY {
  font-family: Arial, Verdana, sans-serif; /* Рубленый шрифт */
  font-size: 90%; /* Размер шрифта */
  color: navy /* Цвет текста */
 }
```

Типы носителей (1/3)



Синтаксис

Синтаксис описания стилей в общем виде представляется следующим образом:

```
selector[, selector[, ...]]
{attribute:value;[atribute:value;...]} /*Перечисление*/
или
selector selector [selector ...]
{attribute:value;[atribute:value;...]} /*Иерархия*/
```

В качестве селектора можно использовать: имя элемента разметки, имя класса и идентификатор объекта на HTML-странице

Комментарии:

~ - отдельных свойств,

/* */ - текстовый блок

Селекторы

Шаблон	Значение	Название
*	Сопоставляется любому элементу.	Универсальный селектор
E	Сопоставляется любому элементу E (т.е. элементу типа E.	Селекторы типа
E F	Сопоставляется любому элементу F, который является потомком элемента E.	Селекторы потомков (контекстные)
E > F	Сопоставляется любому элементу F, который является дочерним элементом элемента E.	Селекторы дочерних элементов
E:first-child	Сопоставляется элементу E, если он является первым дочерним элементом своего родительского элемента.	Псевдокласс :first-child
E:link E:visited	Сопоставляется элементу E, если он является привязкой гиперссылки, направляющей к документу, которой еще не был просмотрен (:link) или уже был просмотрен (:visited).	Псевдоклассы ссылок

Селекторы (1/2)

Шаблон	Значение	Название
E:active E:hover E:focus	Сопоставляется элементу E во время определенных действий пользователя.	Динамические псевдоклассы
E:lang(c)	Сопоставляется элементу E, если он присутствует в разговорном языке (язык документа указывает, каким образом определяется разговорный язык).	Псевдокласс :lang
E + F	Сопоставляется любому элементу F, которому непосредственно предшествует элемент E.	Селекторы сестринских элементов
E[foo]	Сопоставляется любому элементу E с набором атрибутов "foo" (независимо от значения).	Селекторы атрибутов

Селекторы (1/3)

Шаблон	Значение	Название
E[foo~="warning"]	Сопоставляется любому элементу E, у которого значением атрибута "foo" является список значений, разделенных пробелами, и одно из этих значений в точности равно "warning".	Селекторы атрибутов
E[lang|="en"]	Сопоставляется любому элементу E, атрибут "lang" которого имеет список значений, разделенных знаками дефиса, начинающийся (слева) со значения "en".	Селекторы атрибутов
DIV.warning	Значение аналогично значению DIV[class~="warning"].	Селекторы классов
E#myid	Сопоставляется любому элементу E, атрибут ID которого равен "myid".	ID-селекторы

Селекторы (1/3)

```
H1 {
font-family: Arial, Helvetica, sans-serif;
font-size: 160%;
color: #003
}
H2 {
font-family: Arial, Helvetica, sans-serif;
font-size: 135%;
color: #333
}
H3 {
font-family: Arial, Helvetica, sans-serif;
font-size: 120%;
color: #900
}
P {
font-family: Times, serif
}
```

Селекторы тегов

```
H1, H2, H3 {
font-family: Arial, Helvetica, sans-serif
}
H1 {
font-size: 160%;
color: #003
}
H2 {
font-size: 135%;
color: #333
}
H3 {
font-size: 120%;
color: #900
}
```

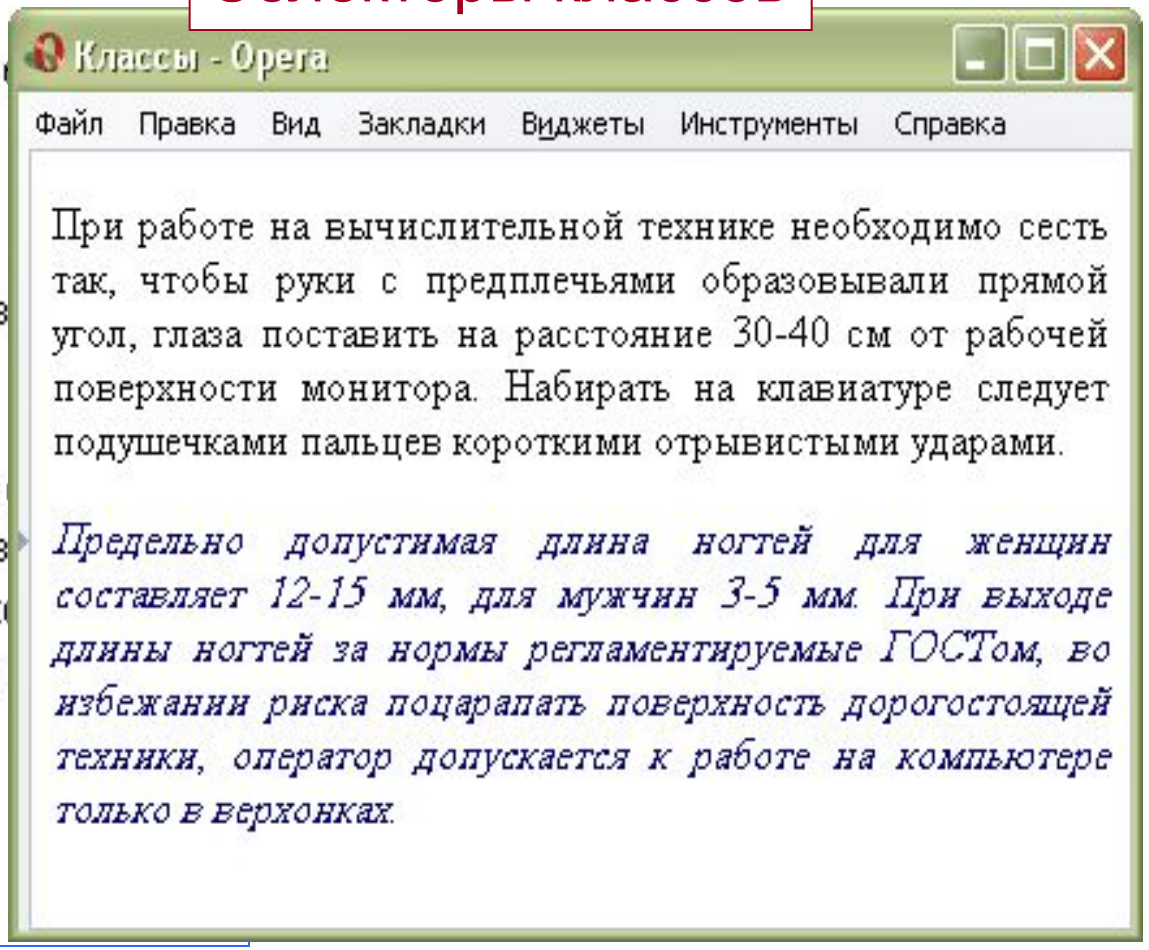
Сгруппированные селекторы

Селекторы (1/4)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DT
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type
<title>Классы</title>
<style type="text/css">
P { /* Обычный параграф */
 text-align: justify; /* Выравнив
}

P.cite { /* Параграф с классом
 text-align: justify; /* Выравнив
 color: navy; /* Синий цвет тек
 font-style: italic; /* Курсивное
}
</style>
</head>
<body>
```

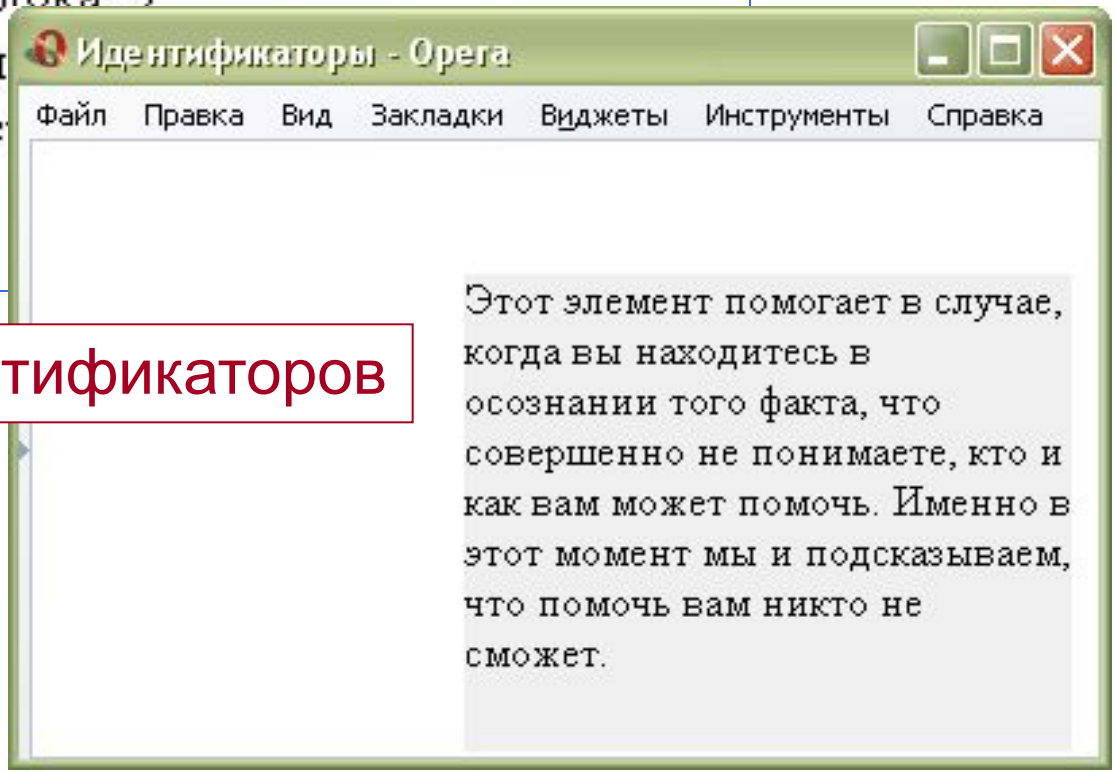
Селекторы классов



Селекторы (1/5)

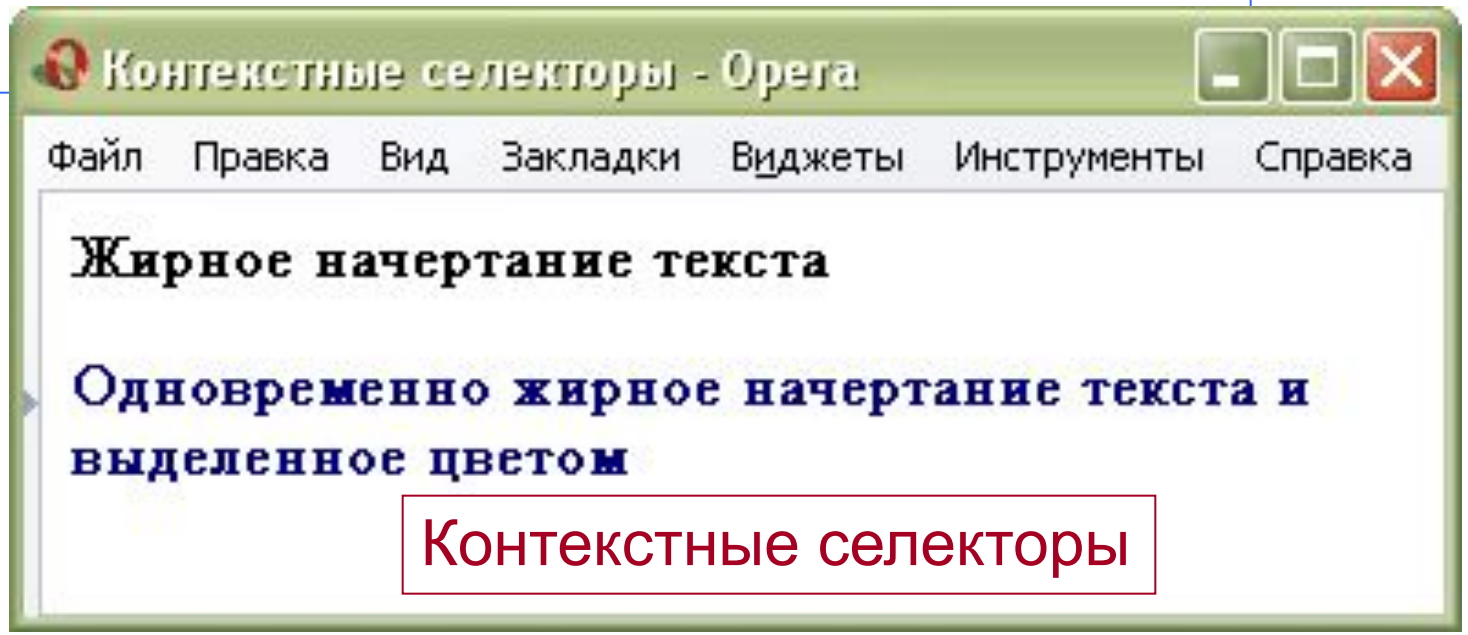
```
<style type="text/css">
#help {
 position: absolute; /* Абсолютное позиционирование */
 left: 160px; /* Положение элемента от левого края */
 top: 50px; /* Положение от верхнего края */
 width: 225px; /* Ширина блока */
 height: 180px; /* Высота бл
 background: #f0f0f0; /* Цве
}
</style>
```

Селекторы идентификаторов



Селекторы (1/6)

```
<style type="text/css">
P B {
 font-family: Times, serif; /* Семейство шрифта */
 font-weight: bold; /* Жирное начертание */
 color: navy; /* Синий цвет текста */
}
</style>
```



Сопоставление шаблонов

- Существующие в CSS принципы сопоставления шаблонов (селекторов) определяют применение правил, задающих стиль, к элементам в дереве документа. Если определенный элемент удовлетворяет всем критериям, устанавливаемым шаблоном, то соответствующий селектор сопоставляется данному элементу.

- Чувствительность имен элементов языка документа к регистру определяется языком документа. В HTML-документах имена элементов не зависят от регистра.

Единицы измерения

in - дюймы	pt - пункты (типограф.)	px - пикселы
cm - сантиметры	pc - пика	em - вычисляется относительно размера шрифта элемента
mm - миллиметры	ex - высота строчной буквы "x" в шрифте	% - проценты

Свойства шрифта

font-family	Используется для указания шрифта или шрифтового семейства, которым будет отображаться элемент. Пример: P {font-family: Times New Roman, sans-serif;}.
font-style	Задает способ начертания шрифта: normal - нормальный (по умолчанию), italic - курсив, oblique - наклонный. Пример: P {font-style: italic;}.
font-variant	Задает варианты начертания шрифта: normal - нормальный (по умолчанию), small-caps - все буквы заглавные).Пример: P {font-variant: small-caps;}.
font-weight	Определяет степень жирности шрифта с помощью параметров: normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, 900.
font-size	Устанавливает размер шрифта. Параметр может указываться как в относительной (проценты), так и абсолютной величине (пункты, пиксели, сантиметры).

Цвет элемента и цвет фона

color	Определяет цвет шрифта элемента.
background-color	Устанавливает цвет фона для элемента. Разные браузеры отображают это свойство по-разному: Microsoft IE отводит под фон элемента всю доступную ширину страницы, а Netscape Navigator – лишь ширину, занимаемую этим элементом.
background-image	Устанавливает или получает фоновый рисунок для элемента. Значения: *none* (по умолчанию), *Url* (задается абсолютный или относительный путь к рисунку).
background-attachment	Устанавливает или получает поведение фонового рисунка для элемента. Значения: *scroll* (по умолчанию, фоновый рисунок прокручивается вместе с элементом), *fixed* (фоновый рисунок не прокручивается вместе с элементом).

Границы (border)

border-width	Ширина границы. Может быть задана значением в пикселях или с помощью зарезервированных слов thin (тонкая), medium (средняя), thick (толстая). Пример: table {border-width: 2px;}.
border-color	Цвет границы. Может принимать значение transparent для задания невидимой, но имеющей ширину, границы. Пример: table {border-color: green;}.
border-style	Задает стиль рисования границы. Может принимать следующие значения: none (по умолчанию), hidden, dotted, dashed, solid, double,groove, ridge, inset, outset. Пример: table {border-style: dashed;}.
border-collapse	Задает стиль рисования таблицы. Может принимать следующие значения: *separate* (по умолчанию) - Ячейки отделены друг от друга *collapse* - ячейки не имеют промежутков между собой.

Границы (1/2)

Пример стиля groove

Верхняя часть заголовка зеленого цвета (dotted)

Верхняя граница окрашена в синий цвет	Пример стиля dashed

Пример стиля double	Граница из двух линий. Сумма ширины каждой линии и промежуток между ними равняется значению border-width.

Пример использования тега DIV

Пример атрибутов
границы элемента DIV

Это обычный вид таблицы	Ячейки отделены друг от друга

Курсоры (cursor)

auto	По умолчанию. Браузер определяет самостоятельно, какой курсор требуется в данном контексте.
col-resize	Курсор со стрелками влево-вправо и вертикальной разделяющей полоской.
crosshair	Курсор-крест.
default	Стандартный курсор, используемый системой.
hand	Рука с вытянутым указательным пальцем. Используется при гиперссылке.
help	Стрелка с вопросительным знаком.
move	Курсор со 4 стрелками, показывающий возможность перемещения.
no-drop	Рука с перечеркнутым кружочком. Нельзя сбросить объект в текущую позицию курсора.
И др.	

Отступы

margin	Задает или получает ширину отступа от четырех сторон объекта. Пример: H4 {margin:1cm;} body {margin:5px;}
margin-top|left|right|bottom	Задает или получает высоту верхнего|левого|правого|нижнего отступов объекта.
padding	Задает или получает величину пространтства, вставляемого между объектом и его отступом или, если объект имеет границы, между объектом и его границей. Пример: td {padding: 2cm;}
padding-top|left|right|bottom	Задает или получает величину пространтства, вставляемого между объектом и его верхнего|левого|правого|нижнего границей.

Позиционирование и размеры (1/2)

position	Устанавливает или определяет позицию элемента. Значения: static (по умолчанию, позиция объекта определяется текущей разметкой HTML по стандартным правилам), absolute (позиция объекта определяется относительно позиции родительского объекта или относительно объекта **body**, если позиция родителя не определена свойствами top и left), relative (позиция объекта определяется смещением от заданных свойств top и left).
left/top	Устанавливает или определяет позицию элемента относительно левого/верхнего края следующего объекта. Может принимать значения: auto, length (10mm;5px;3em), percentage - число процентов от ширины/высоты родительского объекта (10%).
z-index	Устанавливает или получает порядок слоев для объектов. Значения: *auto* (по умолчанию) *Order* - число, задающее позицию объекта в слоях.

Позиционирование и размеры (2/2)

width	Задаёт или получает ширину элемента
height	Задаёт или получает высоту элемента

Преимущества и недостатки CSS

Преимущества:

- Разделение оформления и содержания;
- Единое оформление документов;
- Централизованное хранение;
- Расширенные возможности;
- Быстрая работа.

Недостатки:

- Браузеры могут по-разному отображать некоторые элементы, а некоторые, вообще, не поддерживать определённые свойства.

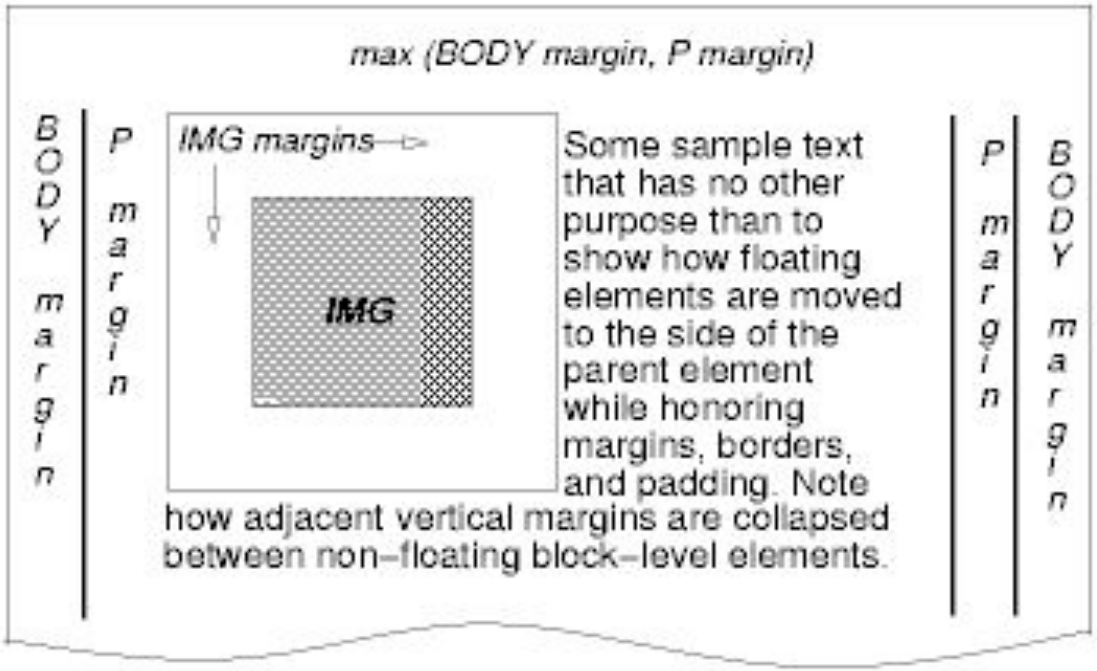
Перемещаемые объекты (1/3)

Следующее правило перемещает все блоки, порожденные элементом IMG с class="icon", влево (и устанавливает значение ширины поля равным нулю):

IMG.icon { float: left; marg

Рассмотрим следующий ис
таблицу стилей:

<!DOCTYPE HTML PUBL
<HTML> <HEAD> <TITL
объекта</TITLE> <STYL
BODY, P, IMG { margin: 2
<BODY> <P><IMG src=i
иллюстрирует перемещ
у которого нет другого...



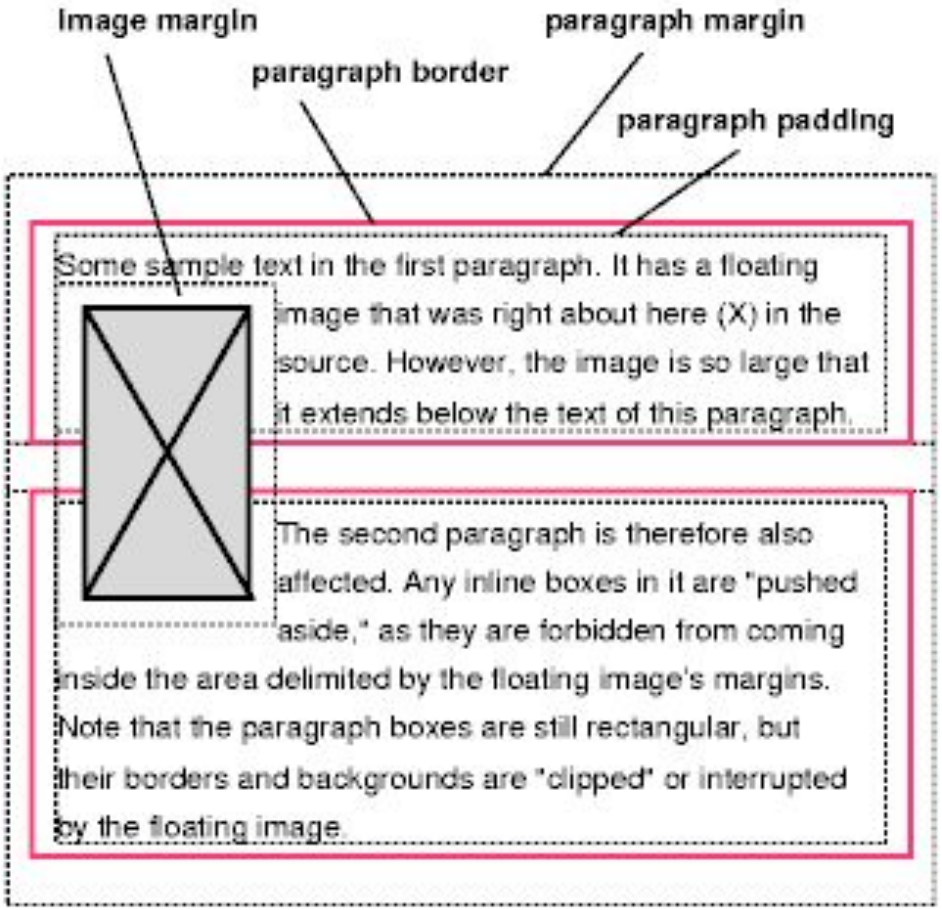
Форматирование было бы
документ выглядел так:

<BODY> <P>Некоторый произвольный текст, <IMG src=img.gif alt=" Данный рисунок иллюстрирует перемещаемые объекты "> у которого нет другого... </BODY>

Объясняется это тем, что содержимое, находящееся слева от перемещаемого объекта, заменяется им, а само отображается справа от него.

Перемещаемые объекты (2/3)

Ниже показано, что происходит при наложении перемещаемого объекта на границы элементов нормального потока.



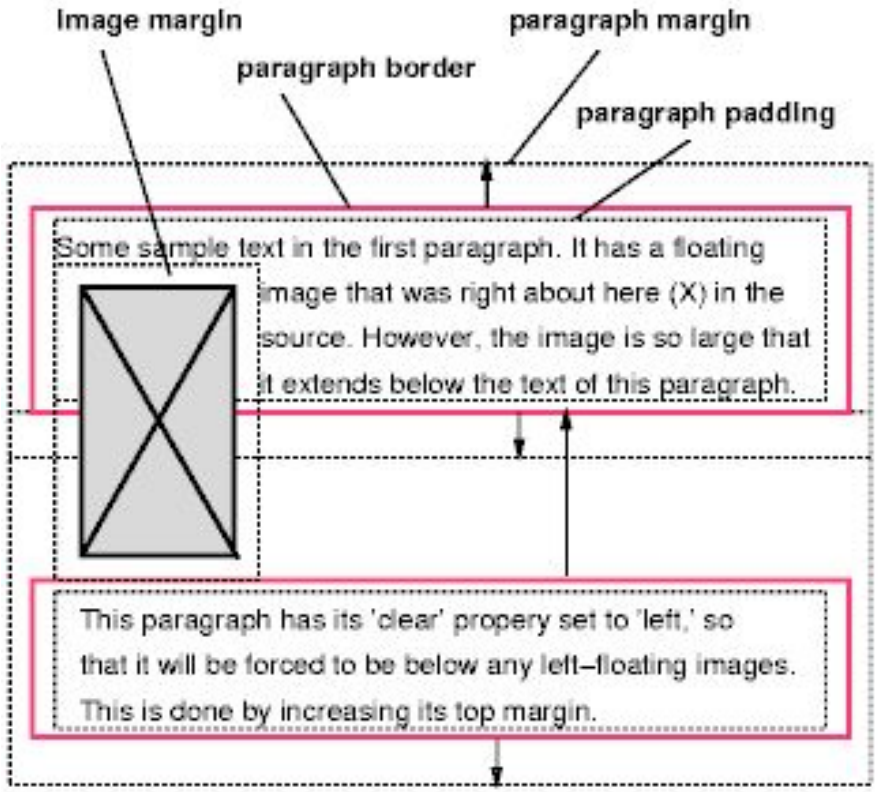
Перемещаемое изображение затеняет границы перекрываемых им структурных блоков.

Перемещаемые объекты (3/3)

В следующем примере проиллюстрировано использование свойства 'clear', позволяющего запрещать перемещение содержимого вдоль относительно объекта.

Правило

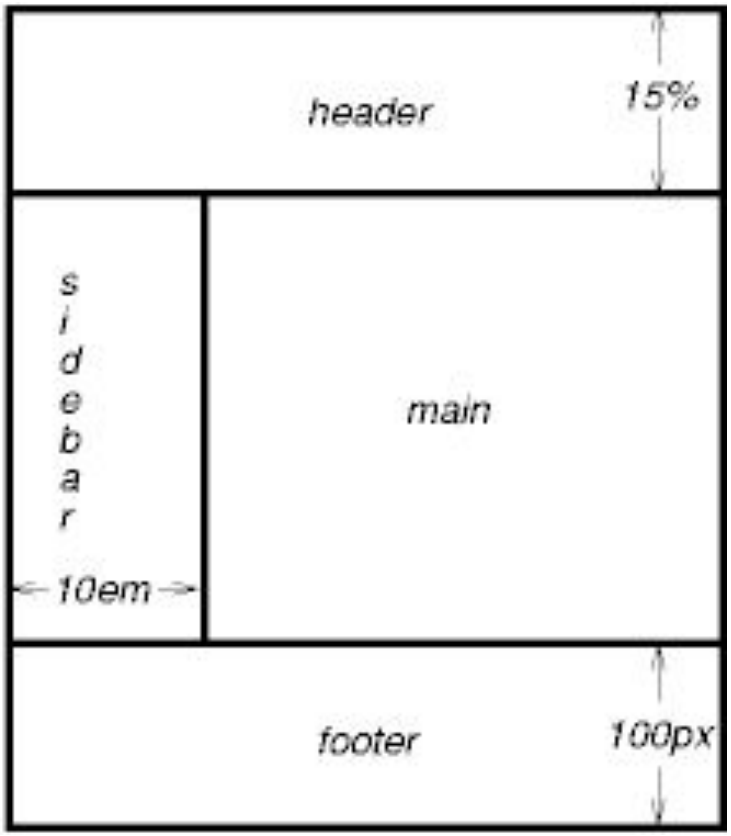
P { clear: left } может привести к следующему форматированию:



В обоих абзацах установлено свойство 'clear: left', при действии которого второй абзац "принудительным образом" располагается ниже перемещаемого объекта. Для этого увеличивается ширина его верхнего поля

Абсолютное позиционирование. Фиксированное позиционирование

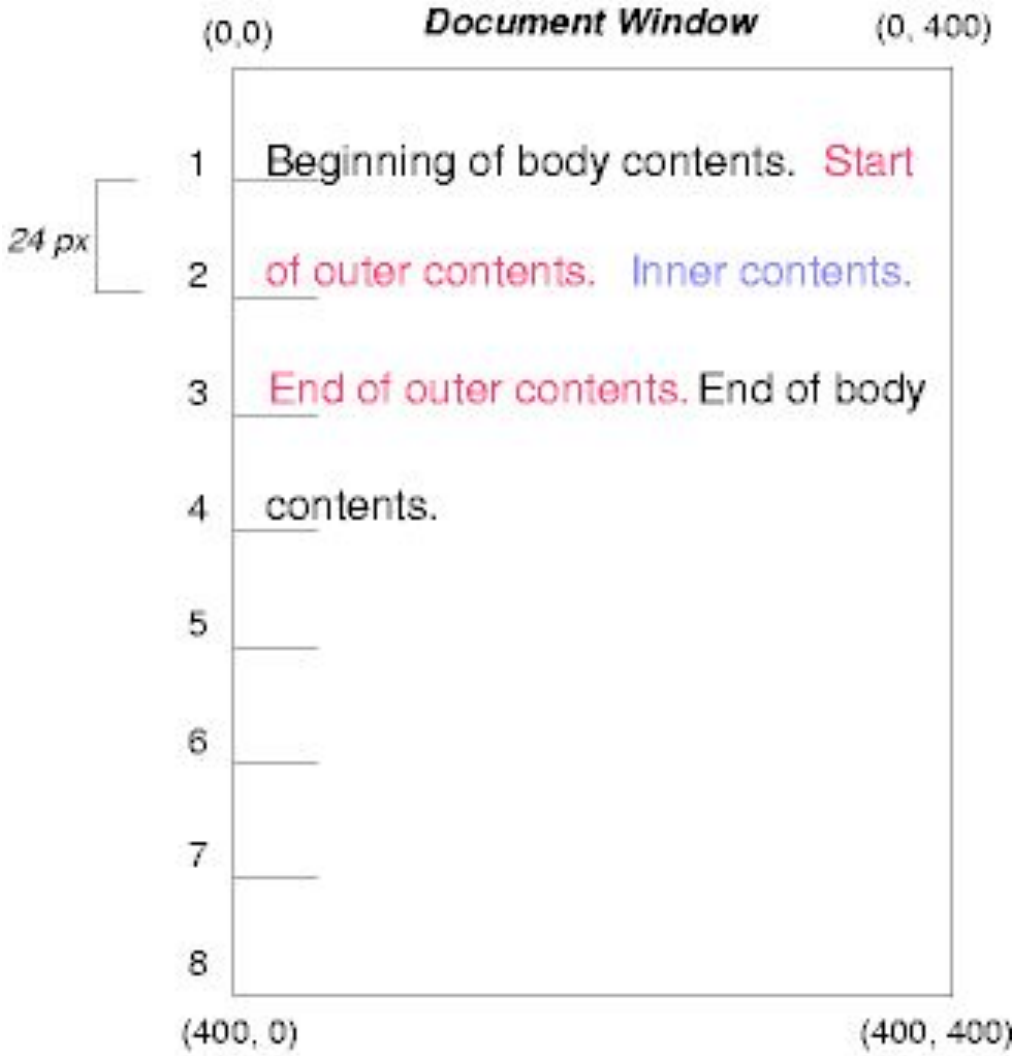
Разработчики могут использовать фиксированное позиционирование для создания презентации в виде совокупности кадров. Рассмотрим один из примеров такой презентации:



Сравнение моделей, описывающих нормальный поток, перемещаемые объекты и абсолютное позиционирование

Нормальный поток

Рассмотрим следующие объявления языка CSS "*outer*" и "*inner*", которые не вносят изменений в нормальный поток блоков: #outer { color: red } #inner { color: blue } Элемент P заключает в себе все содержимое строки: безымянный текст строки и два элемента SPAN. Поэтому все содержимое будет отображаться в контексте строкового форматирования внутри контейнера, порожденного элементом P. В результате получится следующее:



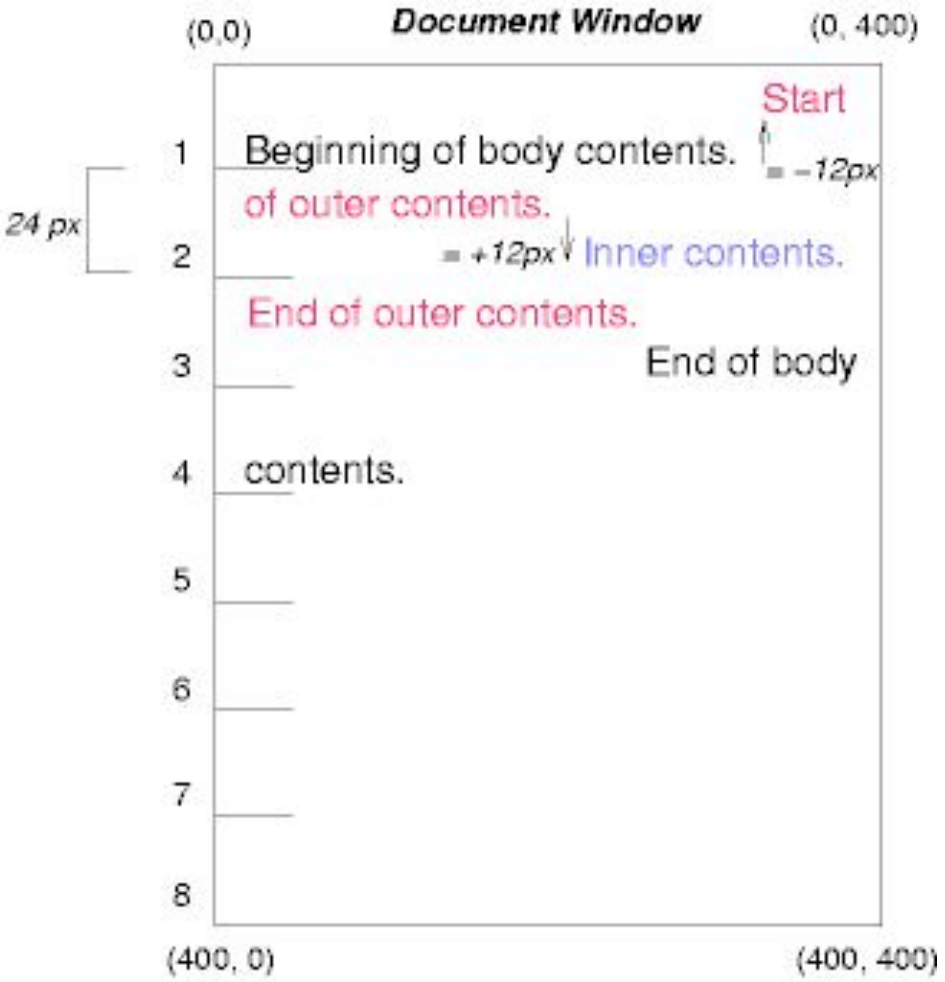
Относительное позиционирование

Чтобы пронаблюдать результаты использования относительного позиционирования, можно воспользоваться следующими правилами:

#outer { position: relative; top: -12px; color: red } #inner { position: relative; top: 12px; color: blue } Текст обычным образом достигает *внешнего* элемента. Затем *внешний* текст принимает те размеры и то положение в конце строки 1, которые назначаются ему в нормальном потоке. Далее строковые блоки, включающие текст (занимающий три строки), перемещаются как единое целое на '-12px'(вверх).

Содержимое *внутреннего* элемента, выступающего в роли дочернего элемента *внешнего*, будет отображено обычным образом после слов "внешнего текста" (в строке 1.5). Однако сам *внутренний* текст смещается относительно *внешнего* на '12px' (вниз) на свое нормальное положение в строке 2.

Обратите внимание, что относительное позиционирование *внешнего* элемента не оказывает никакого влияния на текст, следующий за *внешним* элементом.

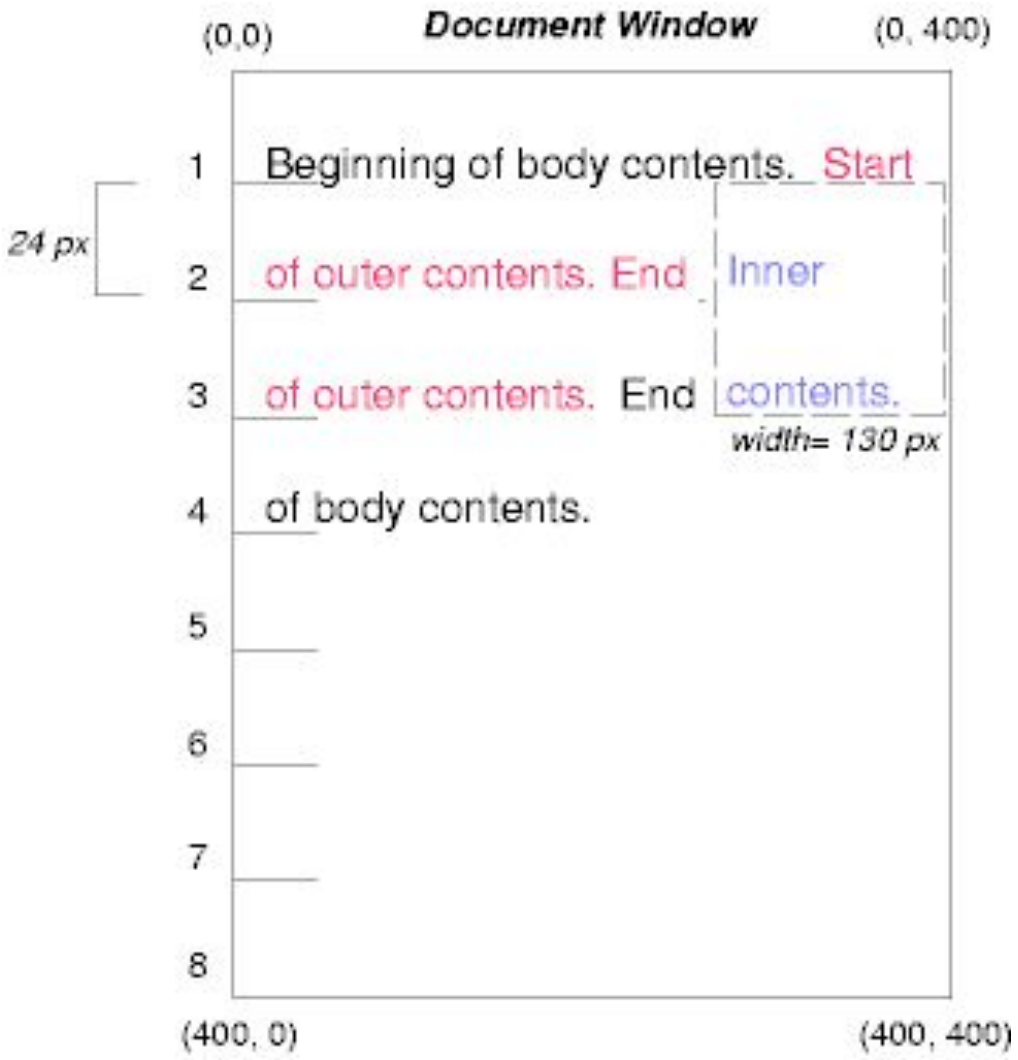


Следует заметить, что если бы *внешний* элемент был смещен на '-24px', то произошло бы наложение содержимого *внешнего* и основного элементов.

Перемещение блоков (1/3)

Рассмотрим результат перемещения текста *внутреннего* элемента к правому краю с использованием следующих правил:

#outer { color: red } #inner { float: right; width: 130px; color: blue } Текст обычным образом выводится до *внутреннего* блока, который изымается из нормального потока и перемещается к правому полю (значение 'width' его ширины было указано явно). Линейные блоки, находящиеся слева от перемещаемого объекта, укорачиваются, и оставшаяся часть текста документа отображается в них.

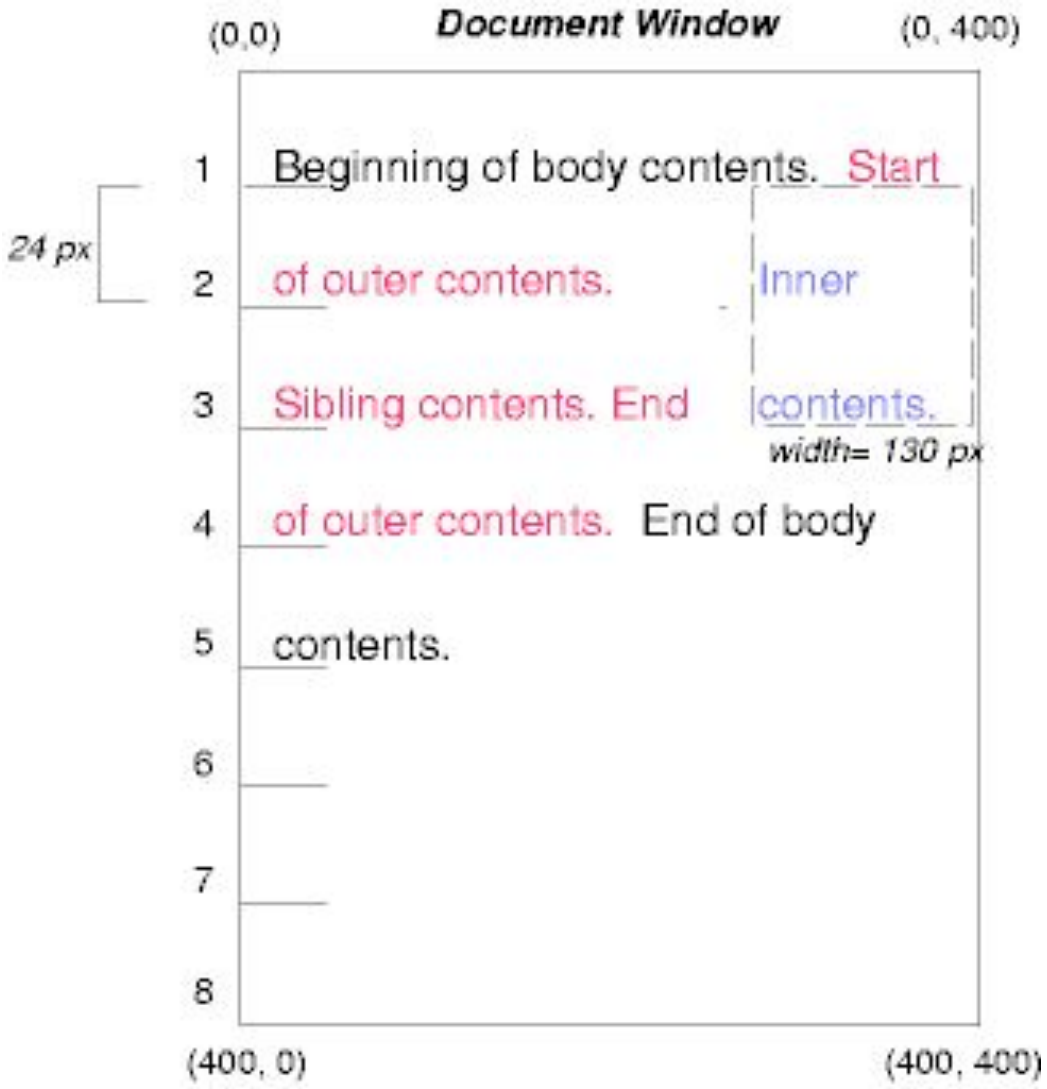


Перемещение блоков (2/3)

Чтобы продемонстрировать эффективность работы свойства 'clear', добавим в пример *сестринский* элемент:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"> <HTML> <HEAD> <TITLE>Сравнение схем позиционирования II</TITLE> </HEAD> <BODY> <P>Начало основного текста. <SPAN id=outer> Начало внешнего текста. <SPAN id=inner> Внутренний текст.</SPAN> <SPAN id=sibling> Содержимое сестринского элемента.</SPAN> Конец внешнего текста.</SPAN> Конец основного текста. </P> </BODY> </HTML> Следующие правила:

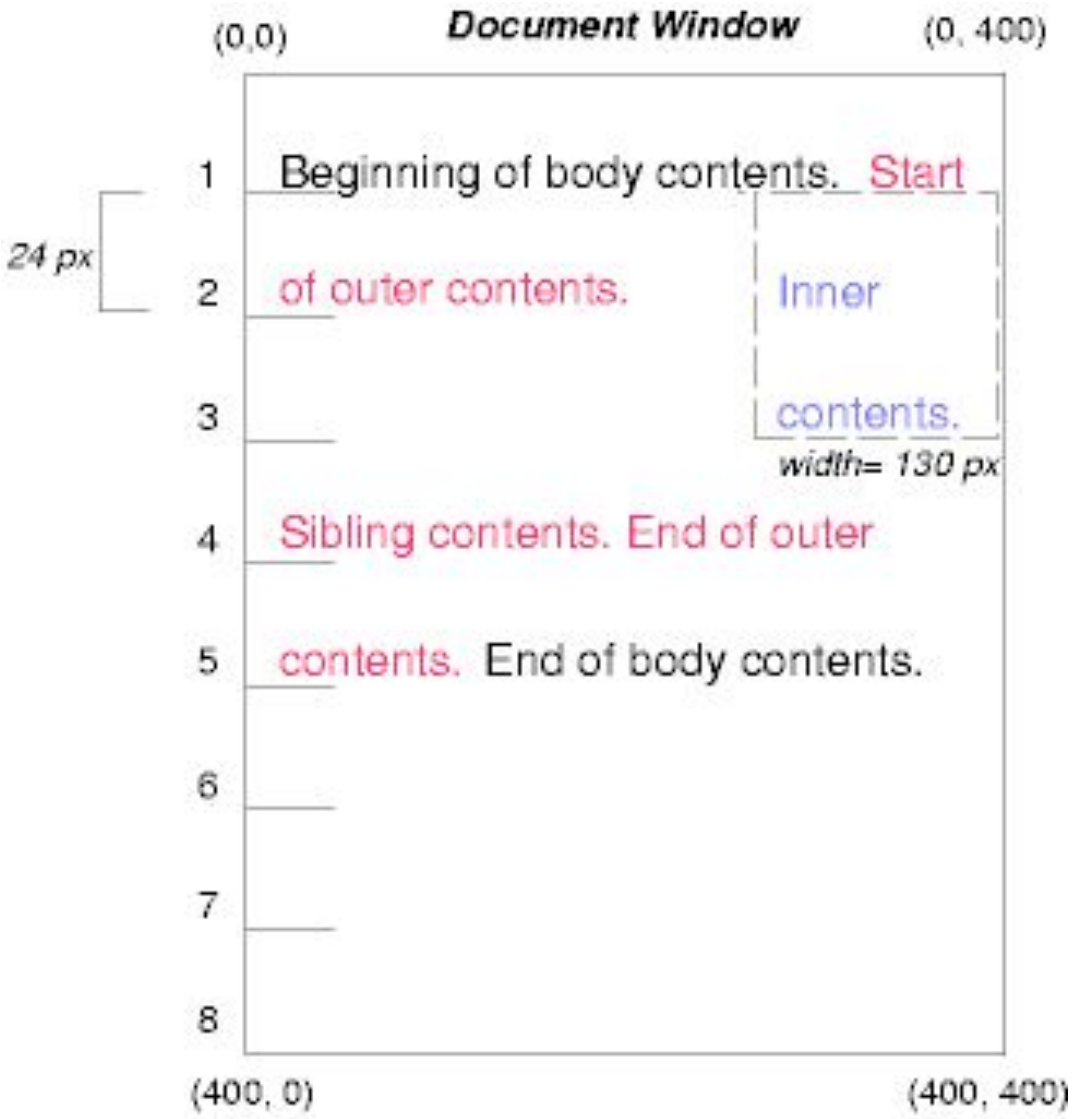
#inner { float: right; width: 130px; color: blue } #sibling { color: red } как и ранее приводят к перемещению *внутреннего* блока к правому краю, а оставшаяся часть текста документа перемещается на освободившееся место:



Перемещение блоков (3/3)

Однако если для свойства 'clear' *сестринского* элемента установлено значение 'right' (т.е. генерируемый *сестринский* блок не располагается следом за перемещаемым блоком справа), то *сестринский* текст будет выводиться ниже перемещаемого объекта:

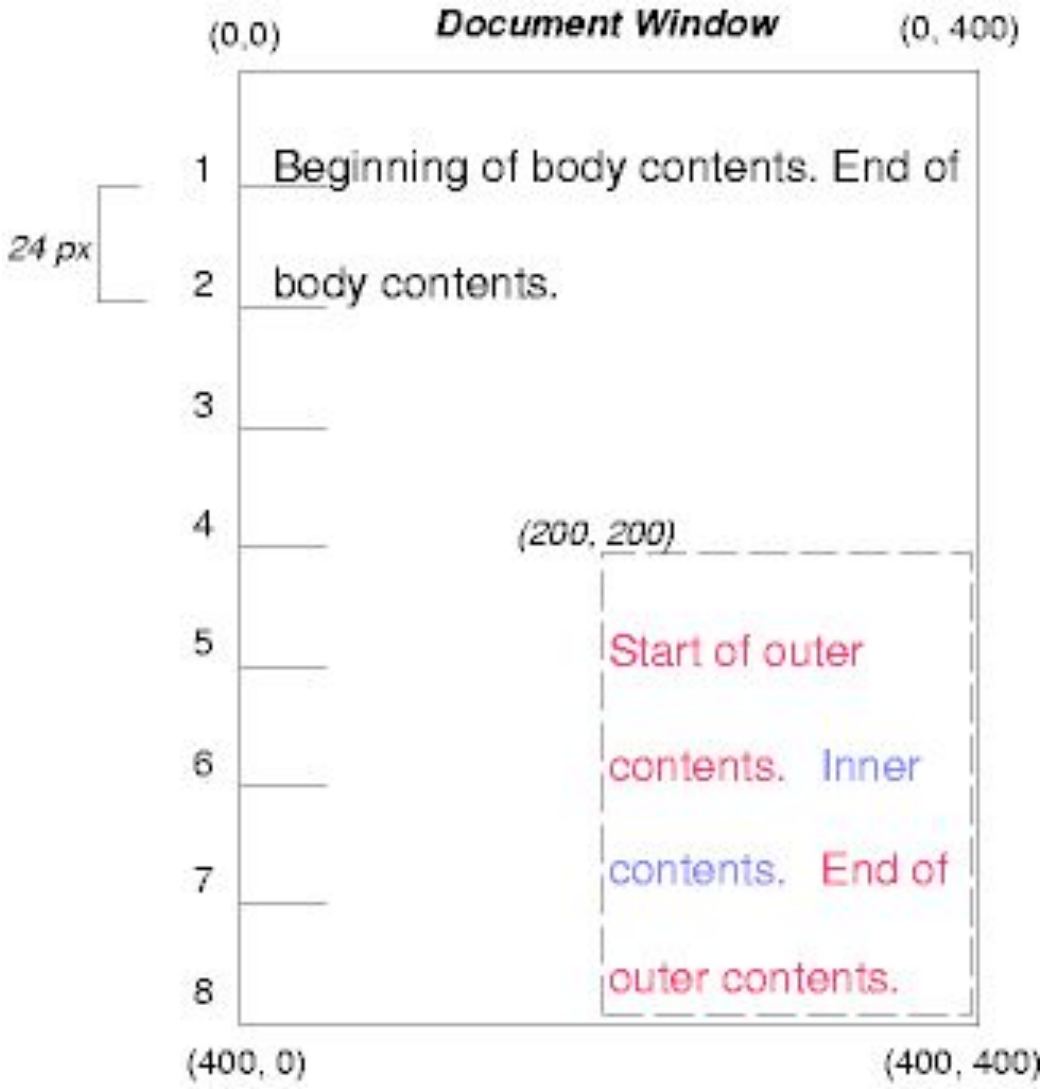
```
#inner { float: right; width: 130px; color: blue }
#sibling { clear: right; color: red }
```



Абсолютное позиционирование (1/4)

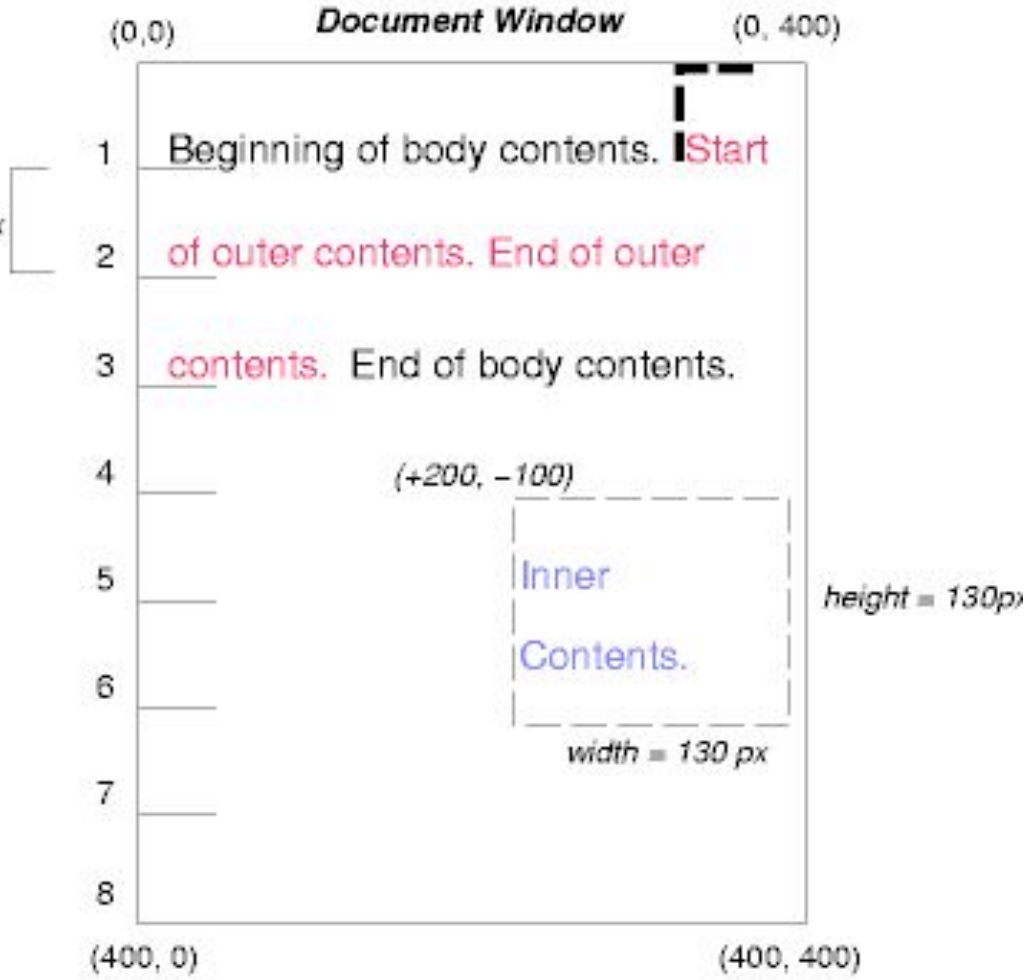
И наконец, рассмотрим результат применения модели абсолютного позиционирования. Рассмотрим следующие объявления *outer* и *inner*:

#outer { position: absolute; top: 200px; left: 200px; width: 200px; color: red; } #inner { color: blue } В результате верхняя часть *внешнего* блока размещается относительно контейнера. Контейнер для размещаемого блока назначается ближайшим размещенным предком (или, в случае отсутствия такового, начальным контейнером, как в данном примере). Верхняя сторона *внешнего* блока находится на '200px' ниже верхней стороны контейнера, а левая сторона на '200px' правее его левой стороны. Дочерний блок *внешнего* блока перемещается обычным образом относительно его родительского блока.



Абсолютное позиционирование (2/4)

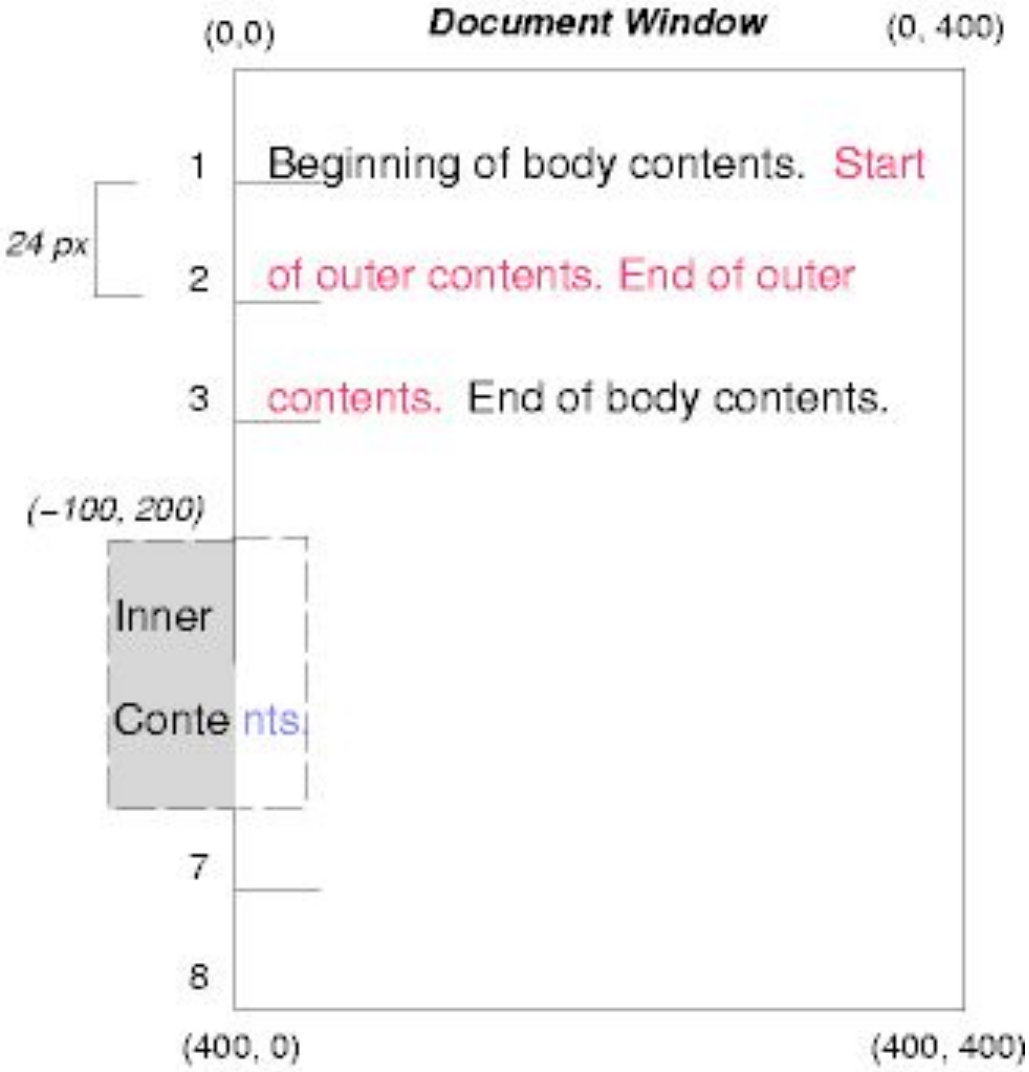
В следующем примере показан абсолютно позиционируемый блок, дочерний по отношению к относительно позиционируемому блоку. Хотя родительский *внешний* блок не смещен, присвоение его свойству 'position' значения 'relative' означает, что он может служить контейнером для позиционируемых потомков. Т.к. *внешний* блок является строковым блоком, разбиваемым на сегменты, распределяемые по нескольким строкам, то в качестве нулевой точки отсчета для смещения 'top' блок является строковым блоком, разбиваемым на сегменты, распределяемые по нескольким строкам, то в качестве нулевой точки отсчета для смещения 'top' и 'left' выступают верхний и левый края первого сегмента (изображенные на рисунке толстыми пунктирными линиями).




```
#outer { position: relative; color: red }
#inner { position: absolute; top:
```

Абсолютное позиционирование (3/4)

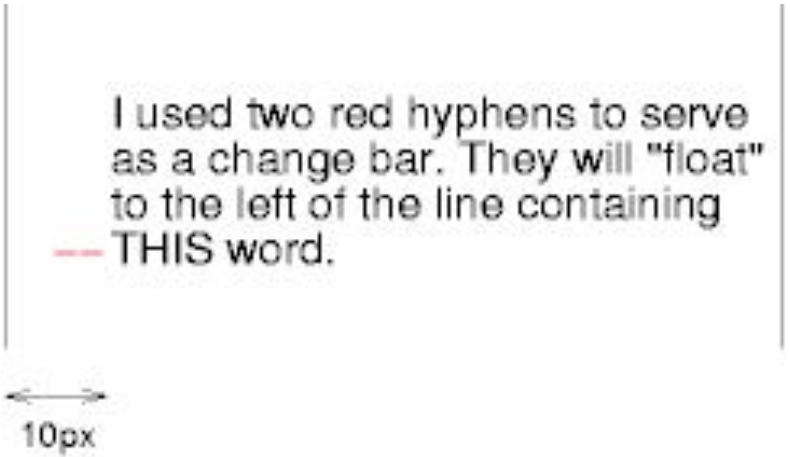
Если *внешний* блок не позиционируется, т.е. используются правила #outer { color: red } #inner { position: absolute; top: 200px; left: -100px; height: 130px; width: 130px; color: blue; } то контейнером для *внутреннего* блока становится начальный контейнер (как в данном примере). На следующем рисунке показано окончательное расположение *внутреннего* блока.



Абсолютное позиционирование (4/4)

Относительное и абсолютное позиционирование может использоваться для вставки маркеров исправлений, как показано в следующем примере. Документ

<P style="position: relative; margin-right: 10px; left: 10px;"> I used two red hyphens to serve as a change bar. They will "float" to the left of the line containing THIS <SPAN style="position: absolute; top: auto; left: -1em; color: red;">--</SPAN> word.</P> приведет к следующему представлению:



Сначала позиционирование абзаца (стороны контейнера которого показаны на рисунке) осуществляется согласно модели нормального потока. Затем он смещается на '10px' относительно левого края контейнера (таким образом, правое поле размером в '10px' было предусмотрено для компенсации этого смещения). Два тире, выполняющие роль маркеров исправлений, изымаются из потока и помещаются на текущей строке (с помощью свойства 'top: auto') на расстоянии '-1em' от левого края контейнера (назначенного элементом P в момент его окончательного позиционирования). В результате создается эффект перемещения маркеров исправлений влево по текущей строке.

Перспективы. CSS3

Некоторые средства, предусмотренные в CSS3:

- Селекторы нового типа, позволяющие форматировать объекты в зависимости от выполнения некоторых условий (например, возможность применять стиль только к первому абзацу, следующему после заголовка).
- Улучшенные средства управление цветом, включая гамма-коррекцию.
- Возможность отображения данных в виде нескольких колонок.
- Новые свойства, предназначенные для создания пользовательского интерфейса.
- Свойства, ориентированные на работу с мобильными устройствами.
- Расширенные средства поддержки масштабируемой векторной графики (Scalable Vector Graphics - SVG).
- Многочисленные новшества, предназначенные для управления размещением фоновых изображений, отображения текстов, форматирования строк и т.д.

Пример использования CSS3

Что нам готовит Firefox 1.5 Firefox 1.5 developer highlights автор: 2005.09.11 Simon Willison перевод: 2005.09.15 Александр Качанов Итак,

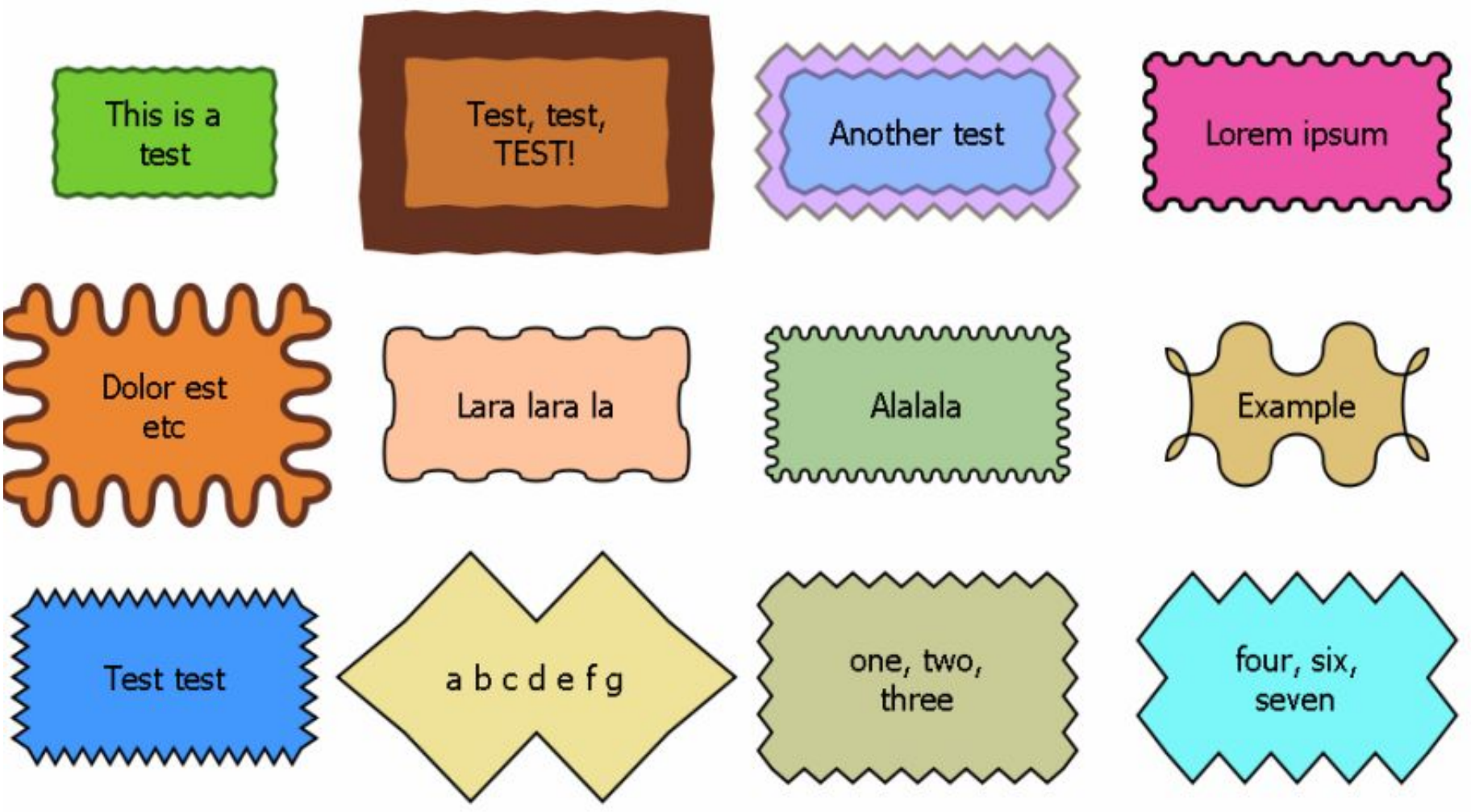
время это самый впечатляющий релиз из всех. В браузер встроен новый генератор страниц Gecko 1.8, в который включено множество

ересных
икций. Новое в
й версии: *
G * Canvas *

Колонки CSS3 * JavaScript 1.6 Рассмотрим каждую функцию поближе. SVG Simon Willison

```
-moz-column-count: 3;
  -moz-border-radius: 5px;
  -moz-column-width: 60px;
  -moz-column-gap: 10px;
  -moz-column-rule: none;
-moz-opacity: 0.6;
```

Пример использования CSS3 (1/2)



примеры\Decorative borders and the Canvas.htm.html

DOM

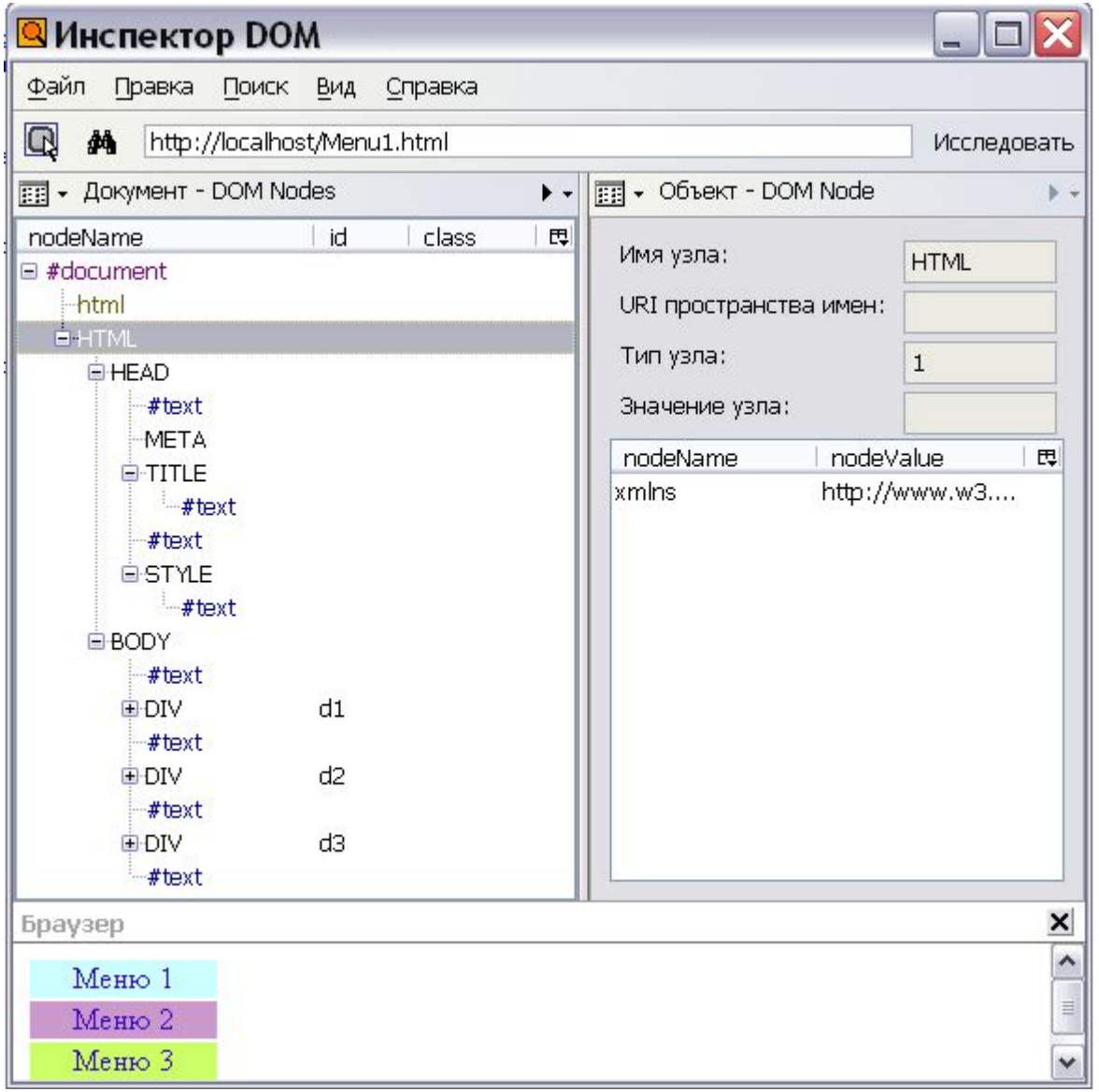
Document Object Model –Объектная модель документа

Определение

DOM – объектная модель документа с соответствующим прикладным программным интерфейсом (API).

У каждого текстового блока имеется свой узел DOM – текстовый узел

В интерфейсе DOM API имеются методы для ввода и удаления отдельных элементов DOM

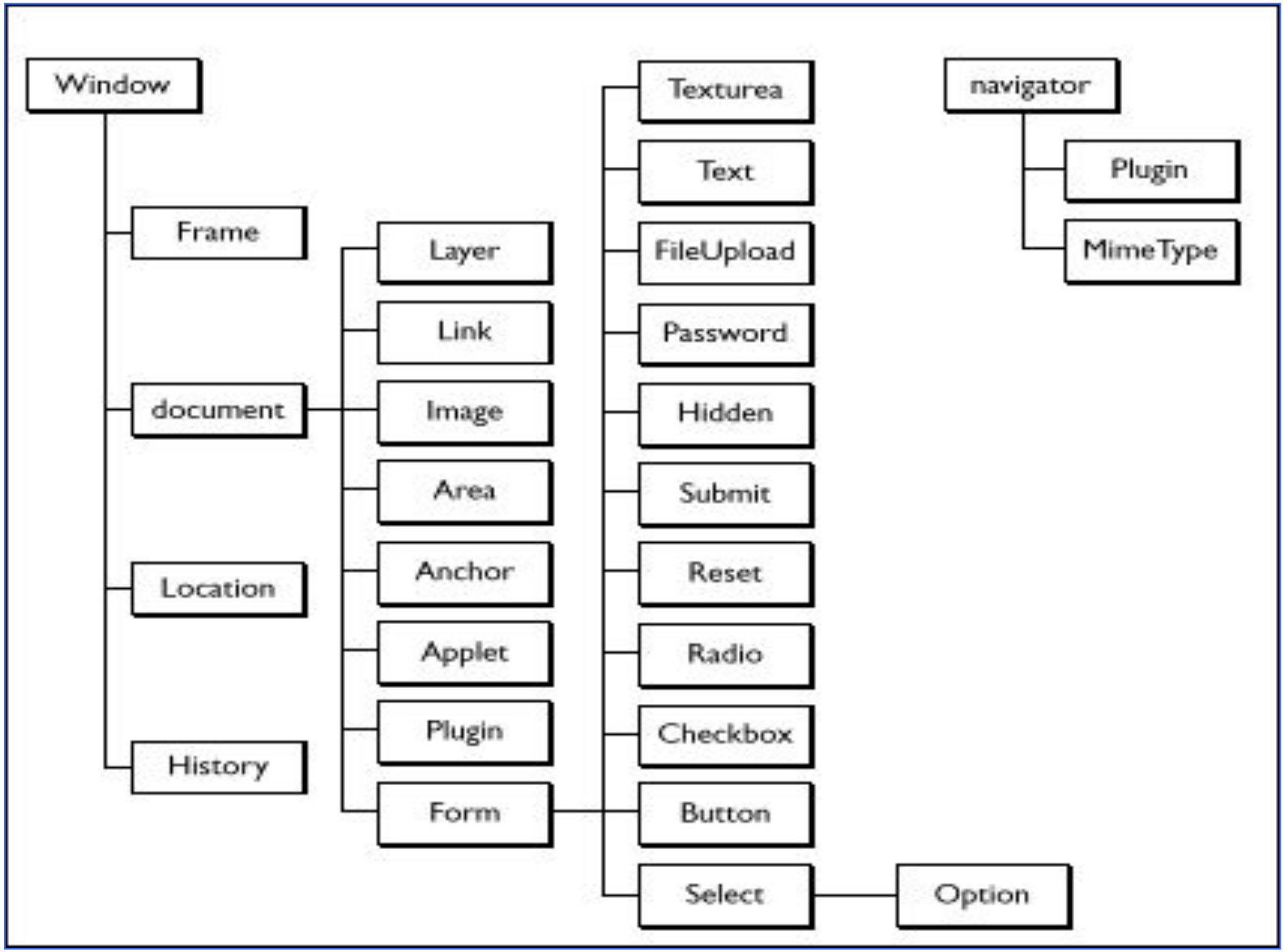


Узлы DOM

Объектную модель документа можно разделить на:

- **DOM:element** – включает набор свойств и методов для всех элементов документа.
- **DOM:window -** объект window и все что с ним связано.
- **DOM:document** – свойства и методы объекта "документ".
- **DOM:event** – набор свойств и методов событий, которые наступают при возникновении какого-либо события вышеперечисленных объектов.
- **DOM:range** – доступ к методам и свойствам областей.
- **DOM:style** – изменения свойств объектов.
- **DOM:selection** – работа с выделенными фрагментами текста, изображений.

Элементы DOM



Достоинства и недостатки

Достоинства:

Удобство использования;

Недостатки:

Сложность

DHTML

Dynamic HTML

(Язык динамической разметки гипертекстов)

Определение

DHTML - это способ создания Web-приложений с использованием HTML, встраиваемого (и выполняемого на стороне клиента) языка JavaScript, CSS и DOM.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
<html>

<head>
<title>Заголовок страницы</title>
<script type="text/javascript">
   window.onload= function () {
    myObj = document.getElementById("navigation");
    // .... какой-то код
   }
  </script>
</head>

<body>

<div id="navigation">
</div>

</body>

</html>
```

Java Script

Язык создания сценариев

Назначение JavaScript

Java Script – клиентский язык создания сценариев. Если браузер поддерживает Java Script, код последнего предоставляет доступ к текущей web-странице и определяет по сценарию свойства клиента, порядок переадресации пользователя и многое другое.

```
<body>
<script language="javascript" type="text/javascript">
window.alert("Welcome!");
</script>
</body>
</html>
```

Java Script и JScript – не одно и то же

Включение кода Java Script

Встроенный в HTML-страницу:

```
<script language="javascript" type="text/javascript">
    //код
</script>
```

Из внешнего файла:

```
<script language="javascript" type="text/javascript"
src="script.js" ></script>
```

Динамическая загрузка:

```
var s = document.createElement("script");
```

Для браузеров без поддержки Java Script:

```
<noscript><!--Welcome to Java Script! //--></noscript>
```

Псевдо-URL и обработчики событий

Код Java Script можно вызвать, используя псевдо-URL:

```
<a href="javascript:window.alert('Hellow!');">Click here</a>
```

Если код, следующий после ‘javascript:’ возвращает результат:

```
<a href="javascript:void(some_func());">Click here</a>
```

Код Java Script можно выполнять с помощью обработчиков событий:

```
<body onload="showText();">
```

При использовании ссылок:

```
<a href="#" onclick="window.alert('Hellow!'); return false;">
Click here</a>
```

Операторы

+	Сложение
-	Вычитание
*	Умножение
/	Деление
++	Приращение на единицу
-	Уменьшение на единицу
-	Унарный минус
%	Взятие модуля

Арифметические

=	Присваивание значения
+=	Увеличение на заданную величину
-=	Уменьшение на заданную величину
*=	Умножение на заданную величину
/=	Деление на заданную величину
%=	Взятие модуля заданной величины

Присваивания

Операторы (1/2). Комментарии

==	Эквивалентность сравниваемых объектов
!=	Не равно
>	Больше
>=	Больше или равно
<	Меньше
<=	Меньше или равно

Операторы сравнения

&&	логическое И
||	логическое ИЛИ
!	логическое отрицание

Логические операторы

Комментарии:

```
// Текст комментариев
/* Текст
комментариев
*/
```

Циклы. Объявление переменных

Цикл For:

for ([start_value;] [condition;] [step]) { program block }

```
for(i=0;i<9;i++) { ... }
```

Цикл while:

while (condition) {program block }

```
wile(j==k) { j++; k--; }
```

Выход из цикла:

break

```
while(i &lt 6) { if(i==3) break; }
```

Продолжение цикла:

continue;

```
while(i &lt 6) { if(i==3) continue; }
```

Оператор объявления переменной:

var variablename [= value | expression];

```
var v1;
v1=1;
v2="var";
```

Циклы. Переменные (1/2)

Переменные языка JavaScript могут хранить значения различных типов:

- **Строки** - последовательность символов;
- **Числовые значения** - целые и действительные числа;
- **Булевы значения** - только два значения **true** или **false**;
- **Массивы** - множества однотипных переменных;
- **Даты** - значения даты и времени.

Строковые переменные

Создать объект String можно одним из нескольких способов:

- присваивание значения при помощи конструктора String();
- использование оператора var и оператора присваивания для создания и инициализации строки;
- создание и инициализация строковой переменной в операторе присваивания;
- преобразование переменной числового типа путем сложения со строковым типом (10+"" ==> "10");

Массивы в JavaScript

Объявление и присвоение переменным значений:

```
ar1 = new Array();
ar2 = new Array(5);
ar3 = (“one”, “two”, “free”)
```

```
var path = "c:/images/" ,
arrayImg = new Array();
arrayImg[0] = path+"img1.gif";
arrayImg[1] = path+"img2.gif";
```

Обращение к элементам массива:

ar[i] – по индексу
ar.length – длина массива
ar.join(“+”) – соединить все элементы в строку, используя разделитель “+”
ar.sort – сортировка массива

```
var path = "c:/images/" ,
arrayImg = new Array(path+"img1.gif", path+"img2.gif");
```

```
var myArray = new Array(3.14, true, 85, date(), "word");
```

Условные операторы

Условный оператор if . . . else:

if (condition); { Программный блок1}
[else { программный блок2 }]

```
<script language ="JavaScript">
today = new date();
minutes = today.getMinutes();
if (minutes >=0 && minutes <= 30)
   document.write("<body text=white bgcolor=blue> Это
написано белым на синем");
     else
       document.write("<body text=red bgcolor=black> Это
написано красным на черном");
</script>
```

Условные операторы (1/2)

оператор ?

(expression) ? trueStatements : falseStatements;

```
<script language ="JavaScript">
var today = new date();
var secs = today.getSeconds();
(secs >=0 && secs <= 30) ?
   document.write("<body text=white bgcolor=blue> Это
написано белым на синем") :
       (secs >=31 && secs <= 50) ?
    document.write("<body text=red bgcolor=black> Это
написано красным на черном"):
   document.write("<body text=red bgcolor=black> Это
написано красным на черном");
</script>
```

Функции в JavaScript

Синтаксис:
function functionName ([arg[, . . .]])
{блок операторов;
[return (value)|value;]
}

Вызов функции:

```
funcName();
s = funcName(“fu”);
window.onload=funcName;
```

```
function retarray() {
  var sarray = new Object();
  sarray[1] = "Java";
  sarray[2] = "Script";
  return (sarray);
}
```

Обращение к аргументам функции при помощи массива argunents[]

```
function showargs(a, b, c) {
arglist = "";
   for (var n=0; n <= arguments.length; n++) {
     arglist += n +"." + arguments[n] + "\n";}
alert(arglist); }
showargs("java","script")
```

Объектная модель JavaScript

Все объекты можно разделить на три группы:

- Объекты браузера (зависимые от браузера объекты: **window** (окно), **location** (местоположение) и **history** (история)).
- Внутренние, или встроенные, объекты языка JavaScript (включают простые типы данных, такие как строки (**string**), математические константы (**math**), дата (**date**) и другие)
- Объекты, связанные с тегами языка HTML (соответствуют тегам, которые формируют текущий документ; включают такие элементы как гиперсвязи и формы).

Методы и свойства объектов

С объектами JavaScript связаны методы, которые позволяют управлять этими объектами, а также, в некоторых случаях, менять их содержимое. Кроме того, в языке JavaScript имеется возможность создавать свои методы объектов. При использовании метода объекта, нужно перед именем метода указать имя объекта, к которому он принадлежит.

```
document.write()
```

~~write()~~

Все стандартные объекты языка JS имеют свойства. Например, bgColor объекта document:

```
document.bgColor
```

Методы объекта Window

- **alert() -** вывод на экран текстовое сообщение;
- **open() -** открытит окна ;
- **close()** – закрыти окна;
- **confirm()** - вывод на экран окна сообщения с кнопками **Yes** и **No (**возвращает ***true*** или ***false***, в зависимости от нажатой кнопки); **prompt() -** выводит на экран диалоговое окно с полем ввода. Метод
- **setTimeout()** устанавливает в текущем окне обработку событий, связанных с таймером; **clearTimeout()** отменяет обработку таких событий.

DOM
(продолжение)

Document Object Model

(Объектная модель документа)

Навигация по дереву документа

Доступ к конкретному элементу (по id)

```
var oList = document.getElementById("components")
```

Доступ по имени дескриптора:

```
var oLiList = document.getElementsByTagName(“LI")
```

Ссылка на родительский элемент

```
var oParent = oList.parentNode
```

Обращение к дочерним элементам:

```
var oItem1 = oList.childNodes[1]
```

```
<DIV> <UL
ID="components">
<LI>HTML</LI>
<LI>CSS</LI>
<LI>Javascript</LI>
</UL> </DIV>
```

Следующий/предыдущий узел того же уровня:

```
oList.firstChild.nextSibling
oList.childNodes[2].previousSibling
```

Свойства-характеристики узлов

- nodeType (только для чтения) – возвращает тип узла (1, 2, 3, 8, 9, 10, 11 для узлов, соответствующих тэгу, атрибуту, тексту, комментарию, документу, DTD, фрагменту, соответственно).

```
oList.nodeType
```

- nodeName (только для чтения) возвращает имя HTML тэга, которому соответствует данный узел, например,P для параграфа илиUL для ненумерованного списка. Для узлов-атрибутов nodeName возвращает название атрибута, а для тестовых узлов возвращает#text.

```
oList.nodeName
```

- nodeValue (только текстовые узлы) - хранит содержание текстового узла. Для элементов возвращает null, а для атрибутов -- значение атрибута

Создание новых узлов

Создание нового элемента определённого типа:

```
var nl = document.createElement(tagName)
```

```
var oItem = document.createElement("LI")
```

Создание текстового элемента:

```
<LI>XML</LI>
```

```
var oText = document.createTextNode(text)
```

```
var oText = document.createTextNode("XML")
```

Редактирование дерева документа

appendChild() – вставка в конец

```
oItem.appendChild(oText); oList.appendChild(oItem)
```

insertBefore() – вставка в определённое место коллекции

```
var oBrother =
oList.firstChild.nextSiblingoList.insertBefore(oItem, oBrother)
```

cloneNode() – клонирование узла

```
var oClone = oList.cloneNode(true)
```

```
<LI>XML</LI>
```

replaceChild() и **replaceNode()** - замещение дочернего/текущего узла:

removeChild() и **removeNode()** – удаление дочернего/текущего узла:

```
var oRemovedItem = oList.removeChild(oList.lastChild)
```

Редактирование дерева документа (1/2)

```
<script language="JavaScript" type="text/javascript">
var nr = 1;
function addItem() {
  var list = document.getElementById("list");
  var newNode = document.createElement("li");
  nr++;
  var newTextNode = document.createTextNode("Item " + nr);
  newNode.appendChild(newTextNode);
  list.replaceChild(newNode, list.firstChild);
}
</script>
<ul id="list"><li>Item 1</li></ul>
<input type="button" onclick="addItem();" value="Replace item"
/>
```

примеры\JavaScript Phrasebook\Chapter 05\replace.html

Работа с атрибутами элементов

createAttribute() – создаёт узел-атрибут

setAttribute() – устанавливает атрибут

```
oNode.id= "newItem" oNode.setAttribute("id","newItem")
```

removeAttribute() – удаляет атрибут

getAttribute() – возвращает значение атрибута

Реагирования на события JavaScript

- С помощью атрибута HTML:
  <body onload="xyz();">
- С помощью атрибута onXXX, доступного в Java Script:
  window.onload = xyz;
- С помощью методов добавления событий:
  attachEvent() – для IE (используется обозначение события "onXXX");

```
узел. attachEvent(type, listener)
```

addEvantListner() – для всех остальных браузеров (используется обозначение события "XXX")

```
узел.addEventListener(type, listener, useCapture)
```

Реагирования на события JavaScript (1/2)

```
<script language="JavaScript" type="text/javascript">
function eventHandler() {
  window.alert("Event fired!");
}
window.onload = function() {
  var button = document.getElementById("eventButton");
  if (button.addEventListener) {
    button.addEventListener("click", eventHandler, false);
  } else if (button.attachEvent) {
    button.attachEvent("onclick", eventHandler);
  }
};
</script>
<input type="button" id="eventButton" value="Click me!" />
```

Удаление обработчиков событий

- Для IE:
  detachEvent()
- Для всех остальных браузеров:
  removeEventListner()

```
узел.removeEventListener(type, listener, useCapture)
```

```
document.removeEventListener("click", mouseClick, true);
```

Всплытие и перехват событий событий

- В IE происходит всплытие событий (событие запускается сначала из того элемента, где оно наступает, а затем оно всплывает вверх по структуре модели DOM).
- W3C (Mozilla, Safari, Konqueror, Opera) события сначала погружаются вниз до целевого элемента, а затем всплывают вверх.

При вводе процесса перехватывания событий в качестве третьего параметра addEventListener можно указать порядок перехвата события.
После перехвата события, его погружение можно прекратить:
В IE – window.event.cancelBuble = false;
В модели W3C – e.stopPropagation.

События

Имя события	Атрибут HTML	Условие возникновения события
Blur	onBlur	Потеря фокуса ввода элементом формы
Change	onChange	Изменение содержимого поля ввода иди области текста, либо выбор нового элемента списка
Click	onClick	Щелчок мыши на элементе формы или гиперсвязи
Focus	onFocus	Получение фокуса ввода элементом формы
Load	onLoad	Завершение загрузки документа

События (1/2)

Имя события	Атрибут HTML	Условие возникновения события
MouseOver	onMouseOver	Помещение указателя мыши на гиперсвязь
MouseOut	onMouseOut	Помещение указателя мыши не на гиперсвязь
Select	onSelect	Выделение текста в поле ввода или области текста
Submit	onSubmit	Передача данных формы
Unload	onUnload	Выгрузка текущего документа и начало загрузки нового

Java Script и CSS

В Java Script имеется возможность задавать любые команды CSS.

В Java Script используется смешанное написание.

Например,чтобы обратиться к свойству font-weight, нужно указать fontWeight.

Доступ к стилям

Доступ к стилям осуществляется через свойство style:

```
<script language="JavaScript" type="text/javascript">
function makeBold() {
  document.getElementById("para").style.fontWeight = "bold";
  window.setTimeout("makeLighter();", 1000);}
function makeLighter() {
  document.getElementById("para").style.fontWeight =
"lighter";
  window.setTimeout("makeBold();", 1000);}
window.onload = makeBold;
</script>
<p id="para">CSS and JavaScript</p>
```

примеры\JavaScript Phrasebook\Chapter 04\style.html

Доступ к классам

Доступ к классам осуществляется через свойство className:

```
<script language="JavaScript" type="text/javascript">
function makeBold() {
  document.getElementById("para").className = "strong";
  window.setTimeout("makeLighter();", 1000);}
function makeLighter() {
  document.getElementById("para").className = "weak";
  window.setTimeout("makeBold();", 1000);}
window.onload = makeBold;</script>
<style type="text/css">
  .strong { font-weight: bold; }
  .weak { font-weight: lighter; }
</style>
<p id="para">CSS and JavaScript</p>
```


\Chapter 04\classname.html

Доступ к отдельным таблицам стилей

Доступ к конкретной таблице стилей можно получить с помощью порядкового номера таблицы стилей в качестве индекса массива stylesheets.

```
<script language="JavaScript" type="text/javascript">
function makeBold() {
  document.styleSheets[0].disabled = false;
document.styleSheets[1].disabled = true;
  window.setTimeout("makeLighter();", 1000);}
…
```

Исчезновение содержимого страницы

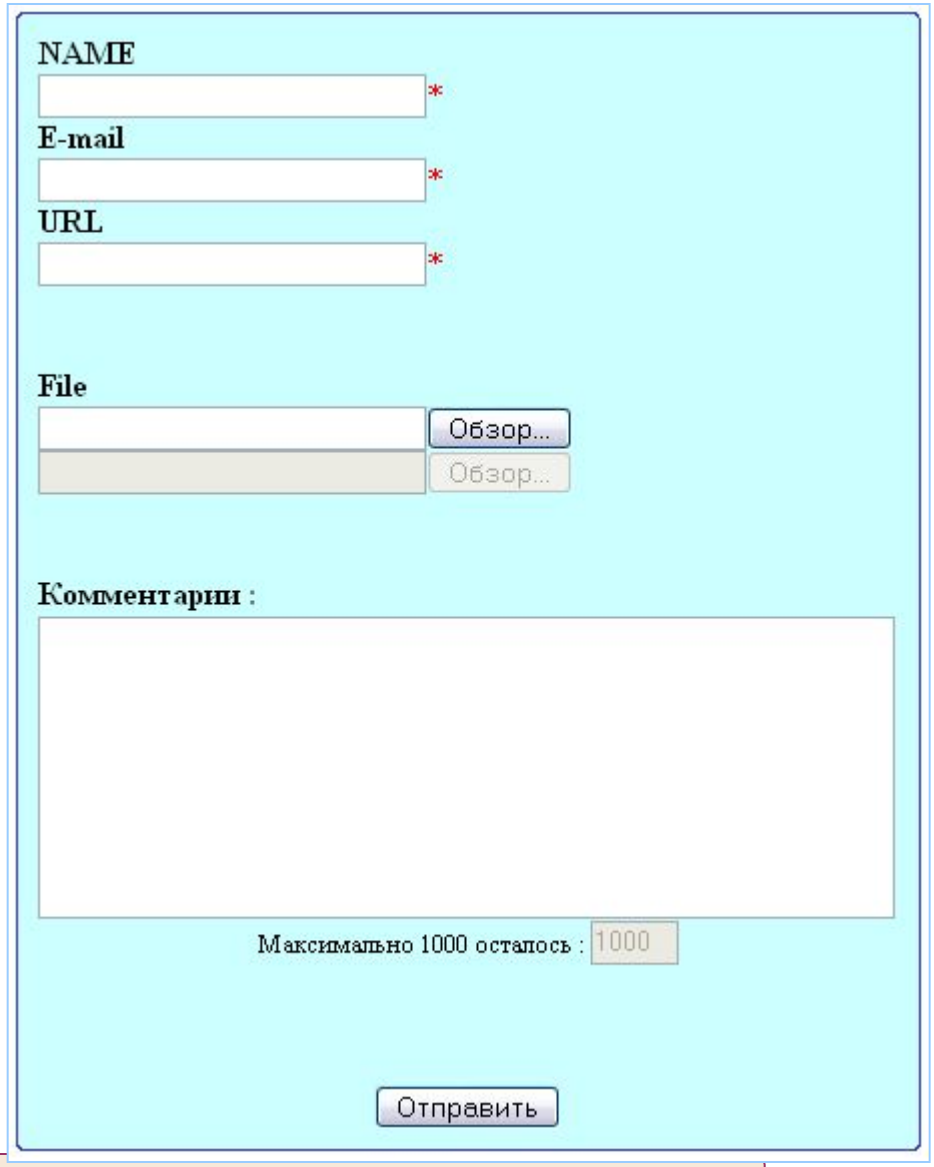
Для этого нужно воспользоваться свойством visibility

```
<script language="JavaScript" type="text/javascript">
  function showHide(show, hide) {
    document.getElementById(show).style.visibility = "visible";
    document.getElementById(hide).style.visibility = "hidden";
  }
</script>
…
```

Что дальше?

- JavaScript 2.0
- CSS 3.0
- HTML 5.0
- XSLT-преобразование (из xml в html)

Задание

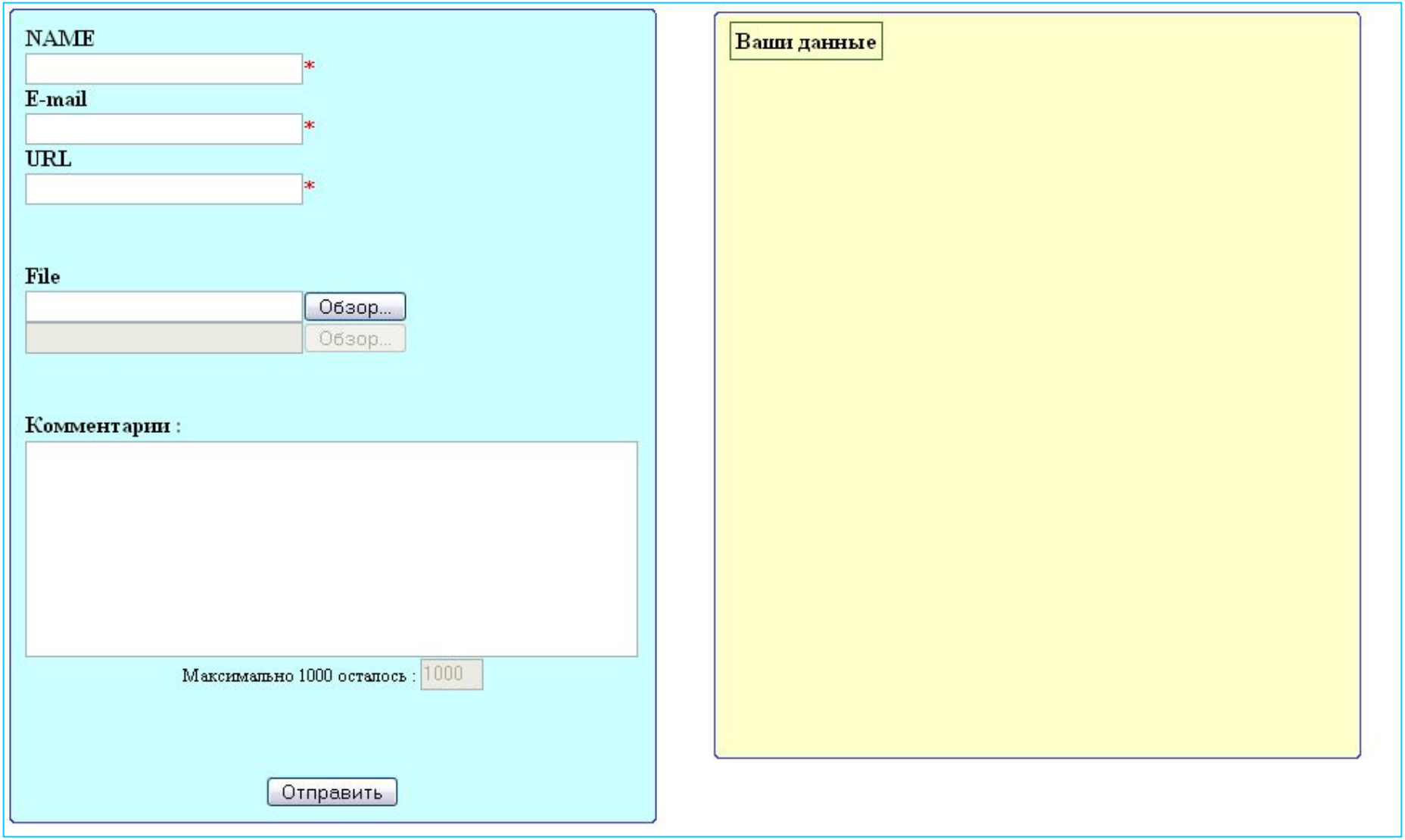


Создать форму, в которой имеется:

-имя пользователя
- e-mail адрес
- URL
- сообщение.
- прикреплённый файл

Так же указывается сколько символов в комментарии можно ещё написать.
При нажатии на «Отправить», проверяется правильно ли заполнена форма (поля, помеченные * должны быть заполнены). Если форма заполнена неправильно, то выдаётся соответствующее сообщение. Иначе, рядом выводится введённая информация, см. далее.

Задание (1/2)


NAME
*
E-mail
*
URL
*
File
Обзор...
Обзор...
Комментарии :
Максимально 1000 осталось : 1000
Отправить
Ваши данные