

Лабораторная работа №9

Статический HTML-документ

Цель работы:

Изучить основы языка разметки гипертекста HTML 4.

Задание:

Создать html-документ, в разметке документа использовать:

- тег для определения кодировки кириллицы `<meta>`;
- тег комментария `<!-- -->`;
- теги форматирования текста: `<p>`, `
`, `<div>`, ``, `<hr>`, `<h1>` ÷ `<h6>`, ``, `<i>`, `<u>`, `<sub>`, `<sup>`, `<pre>`, `<tt>`, продемонстрировать отличия тегов `<p>` и `
`, `<div>` и ``;
- тег для разметки изображения ``;
- тег для разметки гиперссылок `<a>`, разместить ссылки на другой документ, в пределах размечаемого документа, на email;
- с помощью параметров тега `<body>` изменить цвет фона документа, цвет текста, цвета непосещенных и посещенных ссылок документа;

ТЕХНОЛОГИЯ СОЗДАНИЯ HTML ДОКУМЕНТА

Для создания HTML—документов будем использовать текстовый редактор **Блокнот**.

Открываем Блокнот и набираем следующий текст:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Моя первая страница </title>
  </head>
  <body>
    <p>Ура!!! Это 0моя первая страница</p>
```

```
</body>

</html>
```

Сохраните набранный текст в файле с расширением имени .html.

Выберите при Сохранении кодировку utf-8!!!

Откройте созданный файл в браузере и убедитесь, что ваша страничка работает.

Добавьте между тегами <body> и </body> HTML-теги для выполнения задания.

ТЕОРИЯ

ОСНОВЫ HTML

HTML - это язык разметки **документов**. Правильное произношение - **Эйч Ти Эм Эль**.

Вы, наверняка, работали когда-нибудь в редакторе документов Word или подобных офисных приложениях? Наверное вы знаете, что данный вид редакторов имеет богатые возможности для редактирования текста, расположения элементов, вставки картинок и т. д.

Зачем, спросите вы, писать в лекции, посвященной HTML о текстовых процессорах? А вот зачем. Если разобраться, что такое офисный редактор? Это приложение для редактирования и отображения документов.

Ключевое слово здесь - **документ**. То есть, мы создаем, редактируем и просматриваем документ в какой-то программе, в данном случае - в офисном редакторе. Если открыть такой документ в простом текстовом редакторе, например, в Блокноте, то мы увидим множество странных символов и знаков. Эта каша из символов непонятна человечеству, но понятна компьютерам. Благодаря этому внутреннему языку, документ Word приобретает определенную структуру и вид в самом редакторе, а документ предстает перед нами во всей своей красе с отформатированным текстом и картинками на своем месте.

HTML - это язык разметки документов **для браузера**. Word'ом здесь выступает браузер, а документом - HTML страничка. Это самая основа технологии HTML, понимание которой необходимо для того, чтобы не путать язык разметки веб документов с языками программирования. Название говорит за себя - с помощью HTML мы размечаем, где на странице будет показан элемент, картинка или текст, и в каком порядке они будут следовать друг за другом.

Да, простой набор и форматирование текста в офисных приложениях не имеют ни чего общего с программированием. Но наблюдательный читатель заметит важную деталь - в текстовом процессоре мы набираем, редактируем и форматируем текст и картинки с помощью визуальных кнопочек и меню, но почему же HTML код пишется вручную? Зачем изучать так много технических деталей написания разметки для документа?

На самом деле, существует масса редакторов, с помощью которых можно создавать и редактировать HTML странички по аналогии с Word. То есть исходный HTML код для нас скрыт и в него мы не лезем.

Этакий Word для HTML. Такие визуальные редакторы называются:

WYSIWYG редакторы - **What You See Is What You Get**. То есть, если перевести на русский: что видим, то и получаем.

Новички очень часто используют такие редакторы для создания своих первых сайтов. Конечно, это удобно - не нужно углубляться в изучение тегов, стилей оформления и прочих, на первый взгляд, неприятных и сложных вещей. Редактор сам автоматически преобразует наши действия в HTML код.

Но, как говорится, ничего просто так не бывает. А если конкретнее - у такого подхода есть очень серьезные недостатки. Что же мешает всем подряд использовать визуальные редакторы для оформления HTML страничек? Дело в том, что сформированные таким образом страницы имеют, как правило, очень много лишнего кода, очень много ошибок с семантической точки зрения. Сейчас, конечно, нет проблем со скоростным интернет соединением и разница в размере странички в 400 кб и 100 кб не существенна для скорости, однако оптимизированный и правильно написанный HTML код избавляет от множества проблем и дает массу преимуществ, а именно:

- Грамотный HTML код положительно влияет на поисковую оптимизацию, скорость сканирования поисковым роботом сайта. Сгенерированные вузивугой килобайты кода здесь не приемлемы и даже вредны;
- HTML код, сгенерированный WYSIWYG редактором имеет множество семантических ошибок. То есть, теги, генерируемые таким редактором используются не по назначению, там где нужно использовать, например, списки , редактор сгенерирует нам другой, ненужный нам тег. Зависит, конечно, от редактора, но здесь имеются ввиду комплексные решения для создания сайтов, а не простого редактирования текста в текстовой области средствами WYSIWYG.
- Генерируется много лишних тегов и структура документа получается раздутой. Допустим, вы передвигаеете элемент в такой программе сначала вправо, потом влево, потом по центру - от каждого действия остается след в исходном HTML коде. Редактор - это программа и он не может знать, что именно вы хотите получить в результате, он формирует тонны кода с учетом всех возможных вариантов поведения документа в браузере.
- Как правило, редакторы для визуального оформления HTML кода, быстро устаревают. А ввиду отсутствия интереса со стороны профессионалов - вообще лишаются поддержки и останавливаются в развитии. А HTML развивается. Все развивается, кроме вузивуги. Соответственно, они не могут генерировать правильный и современный код, в котором были бы задействованы новые фишки и решения.
- Поддерживать такие проекты и развивать - кара небесная. Об использовании паттернов и повторном использовании кода речи вообще быть не может.

Так что, HTML будем писать только ручками. Адекватных инструментов для визуального редактирования HTML еще не придумали, да и врядли они появятся. Язык разметки HTML прост в освоении и понимании, а средств автоматизации написания HTML кода множество, но об этом в других уроках.

Повозившись немного с WYSIWYG редактором, юные HTML-гуру оставляют это бесперспективное занятие и двигаются дальше.

Структура документа HTML

Мы решили, что код HTML документа будем писать вручную, то есть **верстать**. **HTML Верстка** - процесс создания HTML документа. В простонародье и осведомленных кругах - просто верстка. Любой документ имеет структуру и определенные правила построения. Из каких же элементов состоит код, какая структура у HTML?

Давайте создадим на компьютере первоначальный шаблон - файл **index.html**, откроем с помощью редактора и вставим в него следующий код:

	<code><!doctype html></code>
	<code><html></code>
	<code><head></code>
	<code><title>Заголовок</title></code>
	<code><metacharset="UTF-8"></code>
	<code><linkrel="icon"href="favicon.ico"></code>
	<code><linkrel="stylesheet"href="style.css"></code>
	<code><scriptsrc="script.js"type="text/javascript"></script></code>
	<code></head></code>
	<code><body></code>
	Тело документа
	<code></body></code>
	<code></html></code>

Обратите внимание, документы HTML имеют расширение .html.

Итак, по порядку из примера.

`<!doctype html>` - тип документа (доктайп)

```
<!doctype
html>
```

Данная конструкция всегда указывается в начале документа для правильного "понимания" браузером того, какая версия HTML используется при построении документа.

Ввиду того, что HTML постоянно развивается, он имеет несколько версий, как и любой программный продукт. Текущая версия HTML - пятая и приведенный в примере **доктайп** является актуальным.

В принципе, углубляться в изучение типов документа нет ни какого смысла, ибо с выходом HTML5 данная конструкция стала стандартом. Просто вставляйте ее в начало документа каждый раз, когда начинаете верстать макет сайта.

`<html>` - начало документа

```
<html
>
```

Первый тег, который мы встречаем после доктайпа, это `<html>`.

HTML тег - структурная единица разметки HTML документа. Код HTML складывается из кирпичиков, которые именуются тегами. Каждый тег имеет свою функцию, а изучение языка разметки HTML, в конечном счете, заключается именно в изучении тегов и их свойств в документе.

Хотелось бы отметить, что изучение HTML не такое сложное занятие, как может показаться на первый взгляд. Выучить используемые в разметке документа теги - не такая уж и большая нагрузка на мозг.

Итак, разметка документа начинается с тега `<html>` и заканчивается закрывающим тегом `</html>`. Каждый тег, который содержит в себе другие теги или элементы должен закрываться **закрывающим тегом**. Например, `<html></html>`, `<p></p>`, `<div></div>`, и т. д.

Тег `<html>` является обязательным, так как содержит всю структуру документа и является оболочкой для остальных элементов.

Тег `<head>`

```
<head>
```

Далее, мы видим **тег `<head>`**, который содержит другие, пока не понятные нам элементы. Содержит другие элементы - это значит, что элементы или теги находятся между открывающим и закрывающим тегом конструкции:

	<code><тег></code>
	содержание или другие теги
	<code></тег></code>

Тег `<head>` предназначен для хранения метаданных HTML документа, то есть информации, которая не отображается в самом документе, но является важной и во многом определяет, как документ будет выглядеть и как себя вести.

Данный тег обязателен в документе.

Тег `<title>` - заголовок документа

```
<title>Заголовок</title>
```

Заголовок `<title>` является **обязательным тегом**, содержащим текстовую метаданную, которая отображается в заголовке браузера или вкладки. Тег `<title>` должен находиться в теге `<head>`. Также, содержимое данного тега используется поисковыми системами для отображения документа в результатах выдачи.

Метатег `<meta charset="UTF-8" >`

```
<meta charset="UTF-8">
```

Метатег, содержащий информацию о кодировке документа. Очень желательно указывать данный тег во всех создаваемых документах для правильного отображения. Отсутствие данного тега может привести к тому, что вместо слов, на странице будут отображены неведомые символы и текст перестанет быть разборчивым и

человекопонятным.

Метатег - специализированный тег, предназначенный для предоставления структурированных данных о странице. Метатеги чаще всего используются в теге <head>. Метатеги не являются обязательными в структуре **HTML** документа.

Рекомендую во всех HTML документах изначально использовать кодировку **UTF-8**, как в примере выше.

Фавиконка (favicon)

```
<linkrel="icon"href="favicon.ico">
```

Подключает к документу файл с изображением фавиконки. **Фавиконка (favicon)** - миниатюрный значок, отображаемый рядом с названием документа во вкладке браузера. Фавиконка - это графический файл, размером 16 x 16 (или 32 x 32) пикселей, который может иметь различные форматы, такие, как png, jpg, ico, gif. Традиционно используется формат ico. Анимированные фавиконки - это gif файлы, содержащие анимацию. Наблюдать анимированный фавикон можно, например, ВКонтакте, когда приходит новое сообщение.

CSS стили документа

```
<linkrel="stylesheet"href="style.css">
```

Подключает к документу CSS файл со стилями оформления HTML.

CSS - каскадные стили оформления HTML документа. У каждого тега, который находится в теге <body>, имеется набор свойств, такие как - цвет, ширина, высота, положение относительно других элементов. Все эти свойства и есть стили CSS, которые можно вынести во внешний файл. Конструкция <link> подключает внешние файлы к документу HTML, в том числе и стили CSS.

Примечание: свойство **href** конструкции <link > указывает расположение внешнего файла. В нашем примере, файл style.css и favicon.ico, находятся в той-же папке, что и файл index.html. <link> не имеет закрывающего тега.

Тег <script>

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

Тег <script> содержит код или ссылку на файл **JavaScript** и чаще всего используется внутри тега <head>, хотя инструмент оптимизации скорости загрузки страниц от Google, рекомендует данный тег использовать в конце документа, перед закрывающим тегом </body>.

В нашем примере подключается внешний файл script.js, который находится в той-же папке, что и основной файл index.html.

Итак, друзья, мы рассмотрели основные элементы, которые используются в теге <head> чаще всего. Кроме этих элементов, для <head> есть ряд других, более специфичных и не обязательных.

```
</head>
```

Закрываем тег <head> и шагаем дальше.

Структурные элементы языка

- Единицей языка HTML является **тег** — команда, заключенная в угловые скобки.
- Теги бывают **открывающие** (запускающие действие команды) и **закрывающие** (останавливающие действие команды, соответственно).
- Закрывающий тег отличается наличием прямого слэша `/`.
- Команды тегов распространяются на их **содержимое**:

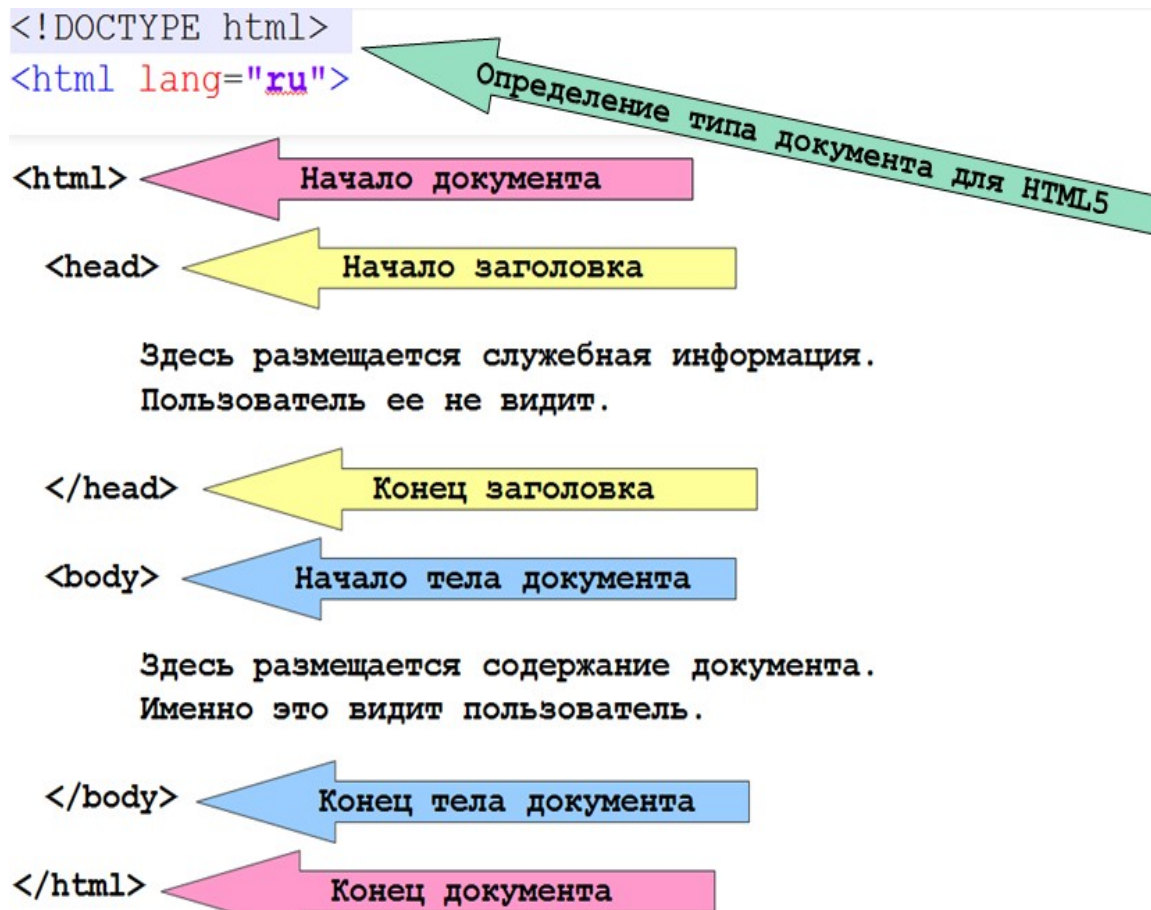


- Целиком пара открывающий-закрывающий тег называется **элементом** или контейнером.
- Помимо элементов, которые содержат и открывающий и закрывающий тег, есть **пустые элементы**, в которых содержание как бы отсутствует, и, соответственно у них нет закрывающего тега. Хотя на самом деле их содержанием является сам открывающий тег.

Пример пустых элементов:

```
<img/>  
<br/>
```

Пример html-документа с разъяснениями:



Еще пример HTML-страницы:

```
1 <html>
2 <head>
3
4 ...Служебная информация...
5
6 </head>
7 <body>
8 <h1>Мой первый HTML документ</h1>
9 <hr>      <!-- горизонтальная линия -->
10 <p>Некоторый текст. Основное содержание текущей страницы. Первый
11 абзац
12 <p>Второй абзац. Для форматирования текста используют разные
13 элементы языка HTML.</p>      <!-- абзац -->
14 </body>
15 </html>
```

На языке html **комментарии** ставятся при помощи символов

```
<!-- содержание комментария строчного-->
<!-- содержание
      комментария
      блочного-->
```

Такой комментарий может быть как строчным, т.е. занимать одну строку документа, так и блочным, занимающим несколько строк.

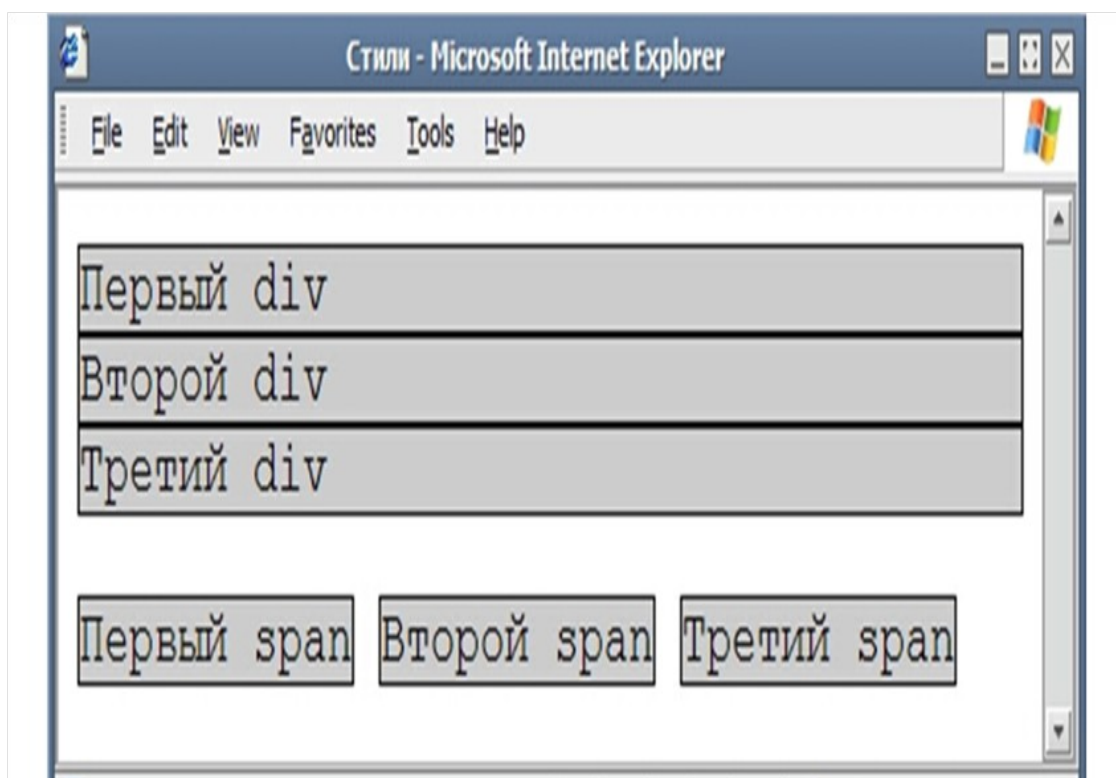
Важно: Для того, чтобы не было проблем с кодировкой символов, необходимо указать тип кодировки в области `head` (т.е. после открывающего тега `head` — головная часть документа)

```
<head>
  <meta charset="utf-8">
  <title>Заголовок окна</title>
</head>
```

Блочные и строчные элементы

Все элементы языка html делятся на **строчные (inline)** и **блочные (block)**. Строчные элементы отличаются тем, что они позволяют разместиться на «своей» строке следующим за ними элементам (позволяют «встать» рядом).

Тогда как блочные элементы займут всю строку, не «пуская» на нее следующие за ними элементы. Схематично это выглядит так:



Элементы форматирования текста

ЗАГОЛОВКИ

- Для размещения заголовков существует тег `<h>` с номером уровня заголовка.

```
<h1></h1>
```

- Самый крупный заголовок соответствует тегу `<h1>`, соответственно заголовок самого низкого уровня (самый мелкий размер шрифта) — `<h6>`.
- Базовый размер шрифта на странице соответствует заголовку `<h3>`:

`<h1>`Заголовок 1`</h1>`

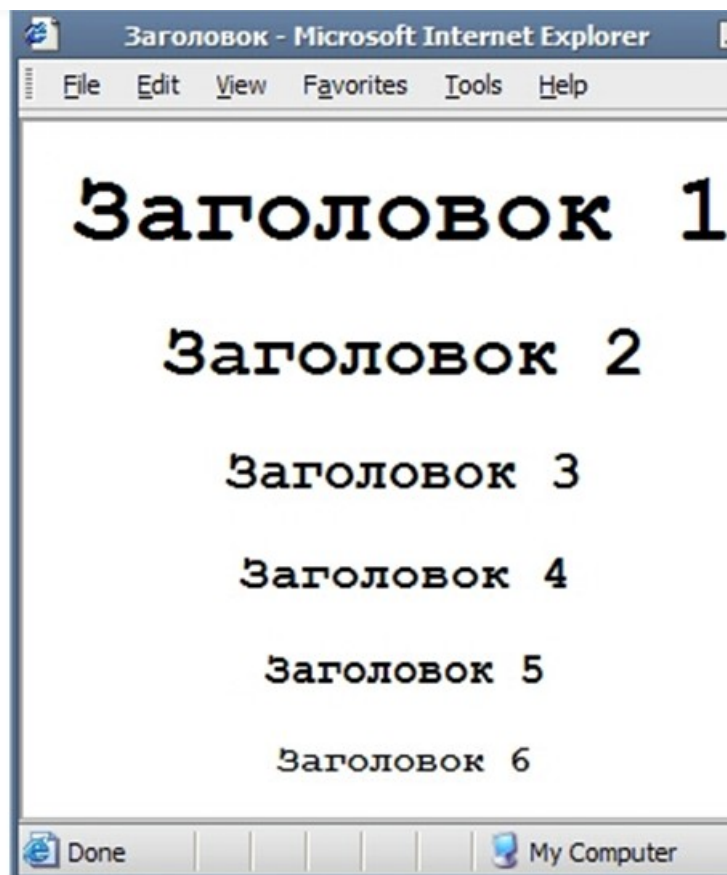
`<h2>`Заголовок 2`</h2>`

`<h3>`Заголовок 3`</h3>`

`<h4>`Заголовок 4`</h4>`

`<h5>`Заголовок 5`</h5>`

`<h6>`Заголовок 6`</h6>`



АТТРИБУТЫ ТЕГА BODY

Для начала рассмотрим два основных атрибута тега `body`:

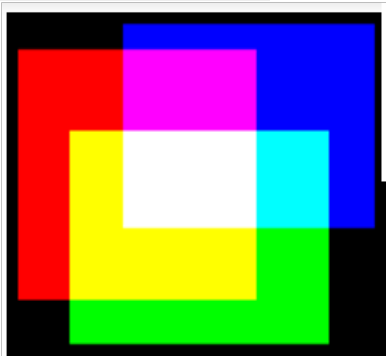
- **bgcolor** — задний фон страницы и
- **text** — цвет текста на всей странице.

Для задания цвета можно использовать названия цветов на английском языке, либо код цвета в шестнадцатеричной системе счисления.

```
<body text="#00ff00">
```

или

```
<body text="green">
```



**Шестнадцатеричная система
счисления :**
0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F.
Диапазон: 00 - FF (0 - 255)
#00FF00 - зеленый.
#FF0000 - красный.
#0000FF - синий.
#FFFFFF - белый.
#000000 - черный.

Перед указанием цвета в 16-й системе обязательно ставится символ «шарп» — `#`
Для подбора подходящего цвета перейдите на страницу [палитры цветов онлайн](#).

Элементы форматирования абзацев

- Для перехода на другую строку текста служит пустой элемент `
`.
- Тогда как для выделения в тексте абзаца служит элемент `<p>`, содержимое которого и является сам абзац. Перед абзацем и после него добавляются отступы, но красная строка при этом не предусмотрена.

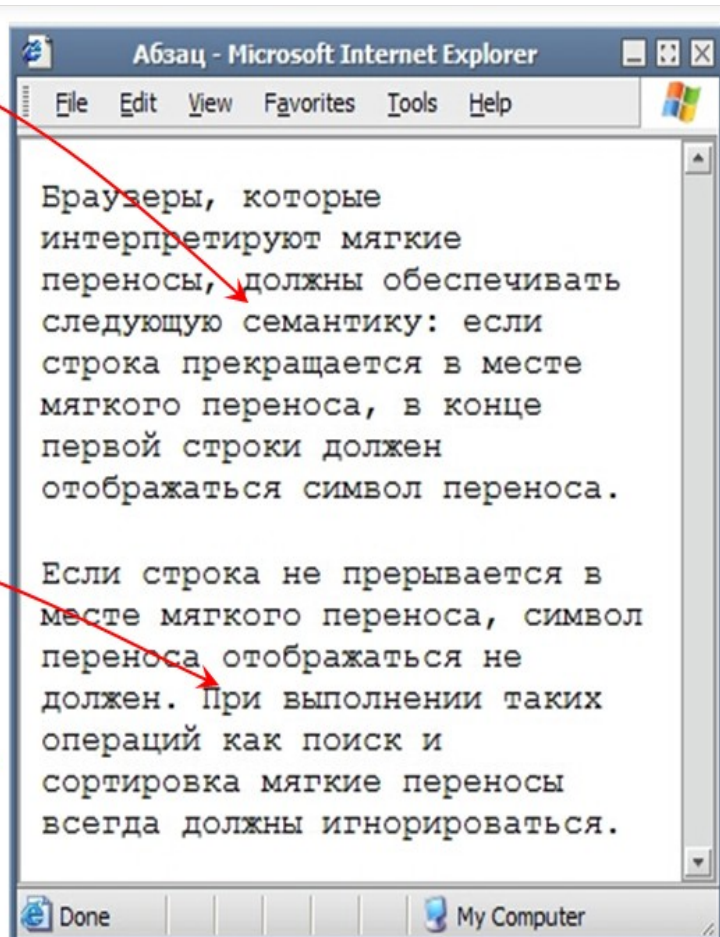
<P>

Браузеры, которые интерпретируют мягкие переносы, должны обеспечивать следующую семантику: если строка прекращается в месте мягкого переноса, в конце первой строки должен отображаться символ переноса.

</P>

<P>

Если строка не прерывается в месте мягкого переноса, символ переноса отображаться не должен. При выполнении таких операций как поиск и сортировка мягкие переносы всегда должны игнорироваться.



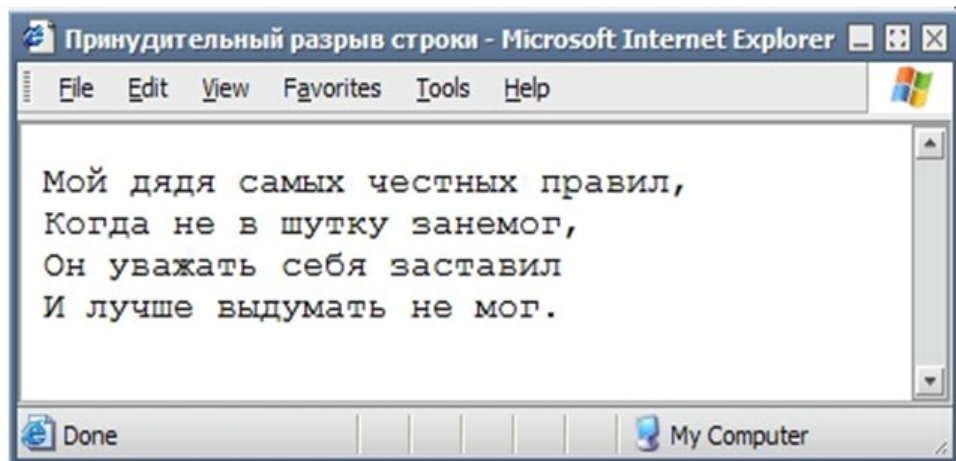
Пример совместного использования тегов
 и <p>:

<p>Мой дядя самых честных правил,

Когда не в шутку занемог,

Он уважать себя заставил

И лучше выдумать не мог.



КУРСИВ, ЖИРНОСТЬ, ПОДЧЕРКИВАНИЕ И ДРУГИЕ ТЕГИ

<i> - курсив

**** - полужирный

<u> - подчеркнутый

<strike> - перечеркнутый

<tt> - моноширинный

<big> - увеличить шрифт

<small> - уменьшить шрифт

<sup> - надиндекс

<sub> - подиндекс

Пример текста тэга **<i>**

Пример текста тэга ****

Пример текста тэга **<u>**

~~Пример текста тэга **<strike>**~~

Пример текста тэга **<tt>**

Пример текста тэга **<big>**

Пример текста тэга **<small>**

Сколько будет 2^3 ?

Формула воды: H_2O

ГОРИЗОНТАЛЬНАЯ ЛИНИЯ

<hr></hr>

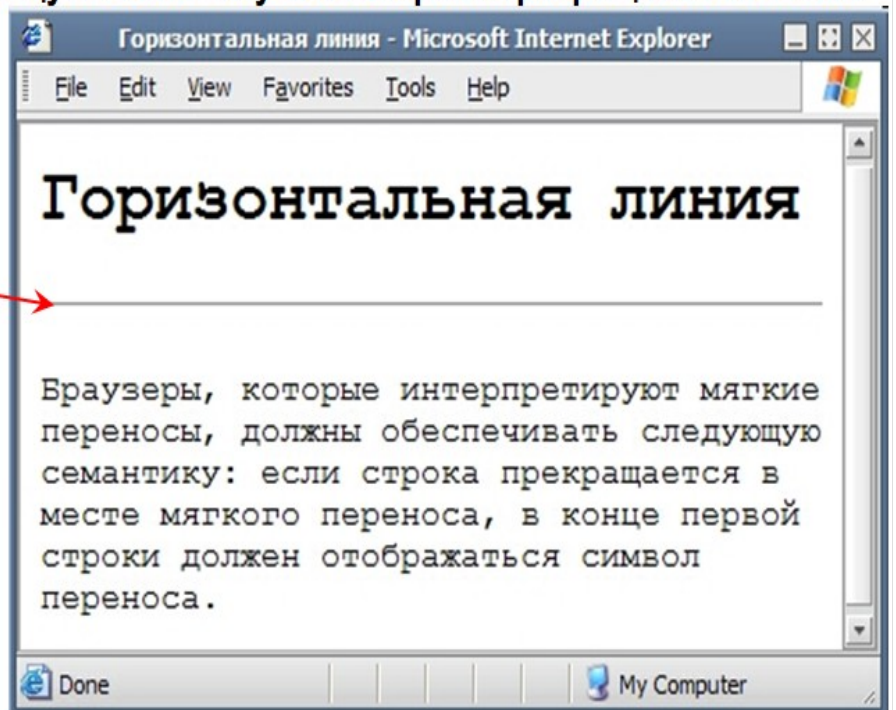
Данный элемент служит для разделения некоторых структурных элементов текста друг от друга. Либо может быть использован просто как эстетический элемент оформления документа:

`<h1>Горизонтальная линия</h1>`

`<hr>`

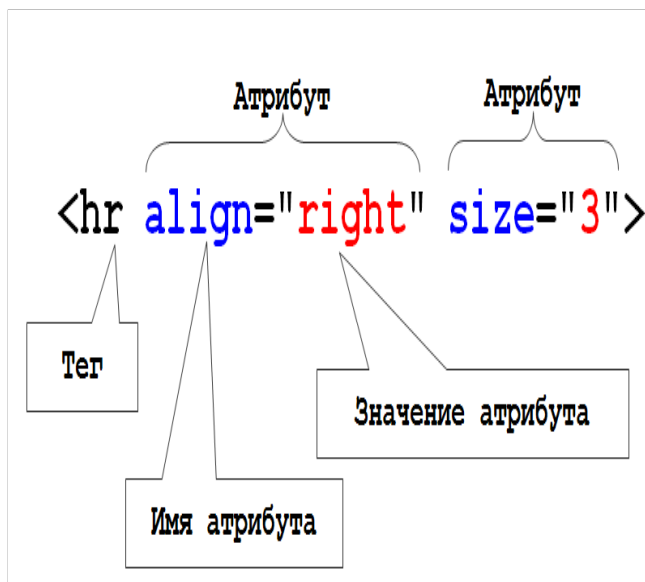
`<p>`

Браузеры, которые интерпретируют мягкие переносы, должны обеспечивать следующую семантику: если строка прекращается в месте мягкого переноса...



Атрибуты тегов

- Для уточнения действия некоторых тегов они дополняются **атрибутами**.
 - Так, у рассмотренного тега горизонтальной линии есть дополнительные свойства, выраженные в атрибутах
 - **size** — ширина линии,
 - **width** — длина линии,
 - **align** — выравнивание линии
- и другие.



- Атрибуты указываются в открывающем теге в виде **атрибут=значение**.
- Атрибутов может быть несколько, тогда они указываются через пробелы, и их порядок следования практически не важен.

`<hr size="3">`

`<hr color="red">`

`<hr noshade>`

`<hr align="right">`

`<hr width="450">`

`<hr size="3" width="50%" align="center">`

.ЦВЕТ И ГАРНИТУРА ШРИФТА

Для форматирования шрифта существует тег ``. Однако, тег уже практически не используется.

``

Тег используется только со своими атрибутами:

- `size` — размер шрифта, от 1 до 7 (3 — базовый размер, 6 — размер заголовка H1)
- `face` — семейство шрифта (`serif` — с засечками, `sans-serif` — рубленый или без засечек, `monospace` — моноширинный)
- `color` — цвет

Пример:

```
<font size="4" color="ff0000" face="Arial, Verdana, sans-serif">
Текст красного цвета, шрифт без засечек
</font>
```

Результат в браузере:

Текст красного цвета, шрифт без засечек

СПЕЦИАЛЬНЫЕ СИМВОЛЫ

код html

©	<code>&copy;</code>	Копирайт
®	<code>&reg;</code>	Знак зарегистрированной торговой марки
™	<code>&trade;</code>	Знак торговой марки
	<code>&shy;</code>	Мягкий перенос
×	<code>&times;</code>	Умножить
÷	<code>&divide;</code>	Разделить
±	<code>&plusmn;</code>	Плюс/минус
≤	<code>&le;</code>	Меньше или равно
≥	<code>&ge;</code>	Больше или равно

