HTML5

Инструктор: Максим

Содержание

1. История развития. HTML

HTML (HyperText Markup Language, язык гипертекстовой разметки) – стандартный язык разметки документов в интернете. Большинство Web-страниц составлены с его помощью. Код страницы **интерпретируется** браузером и отображается в виде документа, в понятной для человека форме

Предполагалось, что язык HTML уйдет в небытие, не дожив до XXI столетия. Организация **W3C** (World Wide Web Consortium, Консорциум Всемирной паутины), которая занимается разработкой и внедрением официальных стандартов Всемирной паутины, забросила язык HTML в далеком **1998** г., считая его не способным на дальнейшее выживание и попытался заменить его языком на основе языка **XML** – **XHTML**

Интерпретатор анализирует и тут же выполняет (собственно интерпретация) программу покомандно (или построчно), по мере поступления её исходного кода на вход интерпретатора. Достоинством такого подхода является мгновенная реакция. Недостаток – такой интерпретатор обнаруживает ошибки в тексте программы только при попытке выполнения команды (или строки) с ошибкой

1. История развития. XML

XML (eXtensible Markup Language, расширяемый язык разметки) – рекомендованный W3C язык разметки. Спецификация XML описывает XML-документы и частично описывает поведение XML-процессоров (программ, читающих XML-документы и обеспечивающих доступ к их содержимому):

```
<?xml version="1.0"?>
<web>
    <theme number="1">HTML</theme>
    <theme number="2">CSS</theme>
    <theme number="3">JavaScript</theme>
    <theme number="4">jQuery</theme>
</web>
```

1. История развития. XHTML

В стандарте **XHTML** (Extensible Hypertext Markup Language, расширяемый язык гипертекстовой разметки) используются те же синтаксические соглашения, что и HTML, но в нем **ужесточены требования** к следованию установленным правилам. Большая часть отступлений от правил разметки, которые сходят с рук в традиционном HTML, попросту неприемлемы в XHTML

Например, разработчик хотел выделить курсивом последнее слово в предложении и должен был написать: `<div>Язык <i>HTML5</i></div>` но ошибся и написал: `<div>Язык <i>HTML5</div></i>`

Когда браузер сталкивается с этой слегка подпорченной разметкой, он в состоянии "понять", что действительно имелось в виду, и без малейших претензий выделяет последнее слово курсивом. Но несовпадающие теги нарушают официальные правила XHTML

1. История развития. HTML5

В **2004** году производителями браузеров: Apple, Mozilla Foundation и Opera Software была организована **WHATWG** (Web Hypertext Appliccation Technology Working Group, рабочая группа по технологии гипертекстовых Web-приложений) по причине отсутствии заинтересованности в HTML и явном пренебрежении к реальным потребностям пользователей со стороны W3C. Данная группа начала работу над **HTML5**

В **2009** году после болезненных размышлений организация W3C решила распустить работающую над XHTML группу и работать вместо этого над формализацией стандарта HTML5

В январе **2011** года WHATWG приняла решение отказаться от упоминания версии HTML5, заменив её простым названием HTML, по которому стандарт определяется по мере его развития

2. Элементы

Документ HTML5, состоит из **элементов**, а элементы состоят из **тегов**. Как правило, элементы имеют открывающий и закрывающий тег (**парный**, контейнер), которые заключаются в угловые скобки:

```
<div>Текст элемента div</div>
```

Текст элемента div

Здесь определен элемент div, который имеет открывающий тег и закрывающий тег . Между этими тегами находится содержимое элемента div. В данном случае в качестве содержимого выступает простой текст "Текст элемента div"

Элементы также могут состоять из одного тега, например, элемент br (тег <br />), функция которого – перенос строки:

```
<div>Текст <br /> элемента div</div>
```

Текст
элемента div

Такие элементы еще называют **пустыми** элементами (void elements)

Как видно, в браузере теги не отображаются, но могут влиять на отображение содержимого

2. Элементы. Синтаксис

Одной из функций механизма представлений HTML5 является обеспечение совместимости нового стандарта с уже существующими HTML/XHTML-документами. Это происходит благодаря наличию трех "режимов" интерпретатора, каждый из которых поддерживает свой синтаксис

Стандартом HTML разрешается набирать теги в любом регистре. Кроме того, можно опускать некоторые закрывающие теги (и косую черту в одиночных тегах). В HTML5 это тоже допустимо, но такой код не будет совместим с синтаксисом XML/XHTML. Обратите внимание, рекомендуется писать код, используя синтаксис XHTML:

Не рекомендуется: `<DIV>оставлять парный тег открытым.`

Рекомендуется: `<div>использовать строчные буквы в именах тегов</div>`

2. Элементы. Синтаксис

Для совместимости со старыми браузерами, которые не понимают синтаксис XHTML, в одиночных тегах желательно оставлять пробел между именем и косой чертой "/":

Не рекомендуется: `<BR>`

Рекомендуется: `<br />`

Парные теги (контейнеры) могут содержать другие теги. При этом действует одно правило – теги должны закрываться в порядке, обратном тому, в котором они открывались:

Не рекомендуется: `<div><i><b> </i></div></b>`

Рекомендуется: `<div><i><b> </b></i></div>`

Вложенные элементы называются **дочерними**, а в которые они вложены – **родителями**. Это относительное свойство, поскольку каждый узел может являться одновременно дочерним для одного и родителем для других. Элемент верхнего уровня (не имеющий родителей) называется **корневым**

2. Элементы. Атрибуты

Любой открывающий или пустой тег может содержать **атрибуты**, определяющие некоторые дополнительные свойства всего элемента. Атрибуты прописываются сразу после имени тега, отделяясь от него и друг от друга пробелом:

```
<a href="http://www.example.com/">Ссылка на сайт</a>
```

Ссылка на сайт

В данном примере элемент "a" (гиперссылка) имеет один атрибут, который отвечает за адрес ссылки. В случаи нажатия на данную гиперссылку, будет произведен переход по указанному адресу

2. Элементы. Атрибуты

Некоторые атрибуты могут иметь только одно допустимое значение и всего два возможных состояния – когда атрибут присутствует в элементе и, соответственно, когда отсутствует. Значение такого атрибута обычно совпадает с его названием. Например, **disabled="disabled"**. Названия в таких атрибутах возможно опускать. Однако для совместимости с синтаксисом XHTML/XML значения указывать все же обязательно. Кроме того, названия всех атрибутов необходимо набирать в нижнем регистре и заключать их значения в двойные кавычки, хотя HTML5 и не ставит такого ограничения

Не рекомендуется: `<input TYPE=text disabled>`
Рекомендуется: `<input type="text" disabled="disabled" />`

2. Элементы. Атрибуты. Глобальные. id. class

Глобальные атрибуты поддерживаются большинством элементов, откуда и получили свое название. С помощью атрибута **id** можно присвоить элементу уникальный идентификатор. Это позволит обращаться к нему из подключенных к документу скриптов, а также задавать правила отображения этого элемента в таблицах стилей:

```
<div id="header">Этому элементу присвоен идентификатор header</div>
```

Похожее назначение у атрибута **class**, с тем отличием, что один и тот же класс может быть присвоен нескольким элементам в документе. Кроме того, одному элементу можно присвоить несколько классов, указав их в значении атрибута через пробел:

```
<div id="first" class="mytext">Элемент класса mytext с идентификатором first</div>
<div class="mytext more">Элемент классов mytext и more</div>
```

Практическое применение этих атрибутов активно начнется после знакомства с таблицами стилей CSS и скриптовым языком JavaScript

2. Элементы. Атрибуты. Глобальные. style. title

Еще одним атрибутом, применяемым совместно с CSS является **style**. В его значении можно напрямую указать CSS-инструкции для отображения элемента:

```
<div style="color: red;">Текст</div>
```

Атрибут **title** определяет элементу всплывающую подсказку, которая будет появляться через некоторое время после наведения на него мыши:

```
<div title="Это текст">Текст с всплывающей
подсказкой</div>
```

2. Элементы. Атрибуты. Глобальные. lang

Текст документа может быть набран как на одном языке, так и содержать вставки на других языках, которые могут различаться по своим правилам оформления текста. Например, для русского, немецкого и английского языка характерны разные кавычки, в которые берется цитата. Чтобы указать язык, на котором написан текст внутри текущего элемента и применяется атрибут **lang**. Браузер использует его значение для правильного отображения некоторых символов:

```
Цитата на французском языке: <q lang="fr">Ce que femme veut, Dieu le veut</q>.<br />
Цитата на немецком: <q lang="de">Der Mensch, versuche die Gotter nicht</q>.<br />
Цитата на английском: <q lang="en">To be or not to be</q>.
```

Цитата на французском языке: «Ce que femme veut, Dieu le veut».
Цитата на немецком: „Der Mensch, versuche die Gotter nicht“.
Цитата на английском: “To be or not to be”.

2. Элементы. Атрибуты. Глобальные. Пользовательские

В отличие от предыдущей версии языка разметки в HTML5 были добавлены **пользовательские** атрибуты (cusom attributes). Теперь разработчик или создатель Web-страницы сам может определить любой атрибут, предваряя его префиксом **data-**:

```
<div data-color="red">Параграф с пользовательским атрибутом</div>
```

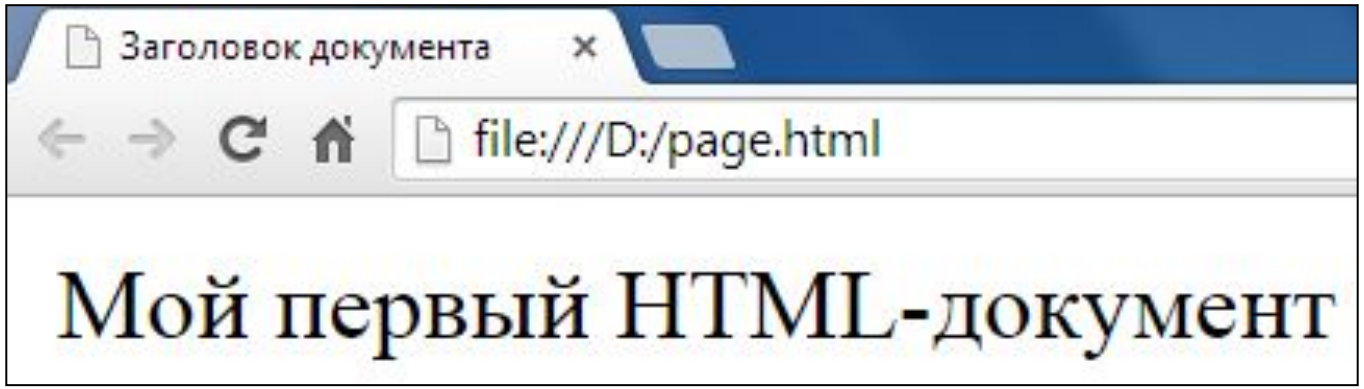
Здесь определен атрибут data-color, который имеет значение "red". Хотя для элемента div в HTML5 не существует подобного атрибута

Данные атрибуты по умолчанию не влияют на работу браузера, они используются вместе с JavaScript и позволяют разработчикам хранить некоторую информацию о элементах страницы

3. Структура HTML-документа

Пример HTML-документа:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Заголовок документа</title>
        <meta charset="utf-8" />
    </head>
    <body>
        Мой первый HTML-документ
    </body>
</html>
```



3. Структура HTML-документа

Первая строчка, элемент **DOCTYPE** предназначен для указания типа текущего документа – **DTD** (Document Type Definition, описание типа документа). Это необходимо, чтобы браузер понимал, как следует интерпретировать текущую Web-страницу

После указания типа следует корневой элемент **html**, охватывающий весь документ. Внутри него один за другим расположены **head** и **body**

head – это "голова" документа, в которой размещается заголовок **title** (его содержимое отображается в заголовке браузера) и прочая служебная информация
body – это "тело" документа, в котором и находится основной текст

Обратите внимание, элементы html, head и title, являются обязательными и должны быть размещены в описанном ранее порядке

Элемент **meta** определяет метаданные документа, чтобы документ корректно отображал текст, необходимо задать кодировку с помощью атрибута **charset**. Рекомендуемой кодировкой является **utf-8**

3. Структура HTML-документа. Форматирования

```
<!DOCTYPE html>
<html>
<head>
<title>Заголовок документа</title>
<meta charset="utf-8" />
</head>
<body>
<p>Тело документа</p>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
   <head>
      <title>Заголовок
документа</title>
      <meta charset="utf-8" />
   </head>
   <body>
      <p>Тело документа</p>
   </body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<title>Заголовок документа</title>
<meta charset="utf-8" />
</head>
<body>
<p>Тело документа</p>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
   <head>
      <title>Заголовок
документа</title>
      <meta charset="utf-8" />
   </head>
   <body>
      <p>Тело документа</p>
   </body>
</html>
```

4. Специальные символы

Иногда возникает необходимость в использовании символа, которого нет на клавиатуре. Как быть в этом случае? Для этого существуют **специальные символы**, состоящие из знаков амперсанда (&), фунта (#), числового кода и точки с запятой (;). Либо из амперсанда, наименования символа и точки с запятой (;)

Некоторые символы:

Символ	Описание	Мнемокод	Код
«	направ. влево двойная угловая кавычка	&laquo;	&#171;
»	направ. вправо двойная угловая кавычка	&raquo;	&#187;
"	двойная кавычка	&quot;	&#34;
&	амперсанд	&amp;	&#38;
<	открывающая треугольная скобка	&lt;	&#60;
>	закрывающая треугольная скобка	&gt;	&#62;
–	тире	&ndash;	&#8211;

5. Элементы группировки. div. p

Элемент **div** служит для структуризации контента на Web-странице, для заключения содержимого в отдельные блоки. Div создает блок, который по умолчанию растягивается по всей ширине браузера, а следующий после div элемент переносится на новую строку:

```
<div>Заголовок документа HTML5</div>
<div>Текст документа HTML5</div>
```

Заголовок документа HTML5
Текст документа HTML5

Параграф создается с помощью элемента **p**, который включает некоторое содержимое. Каждый новый параграф располагается на новой строке:

```
<div>Заголовок документа HTML5</div>
<div>
    <p>Первый параграф</p>
    <p>Второй параграф</p>
</div>
```

Заголовок документа HTML5

Первый параграф

Второй параграф

5. Элементы группировки. pre

Элемент **pre** определяет блок предварительно форматированного текста. Такой текст отображается обычно **моноширинным** шрифтом и со всеми пробелами между словами. По умолчанию, любое количество пробелов идущих в коде подряд, на Web-странице показывается как один. Элемент pre позволяет обойти эту особенность и отображать текст как требуется разработчику:

```
<pre>
Строка №1
    Строка №2
        Строка №3
</pre>
```

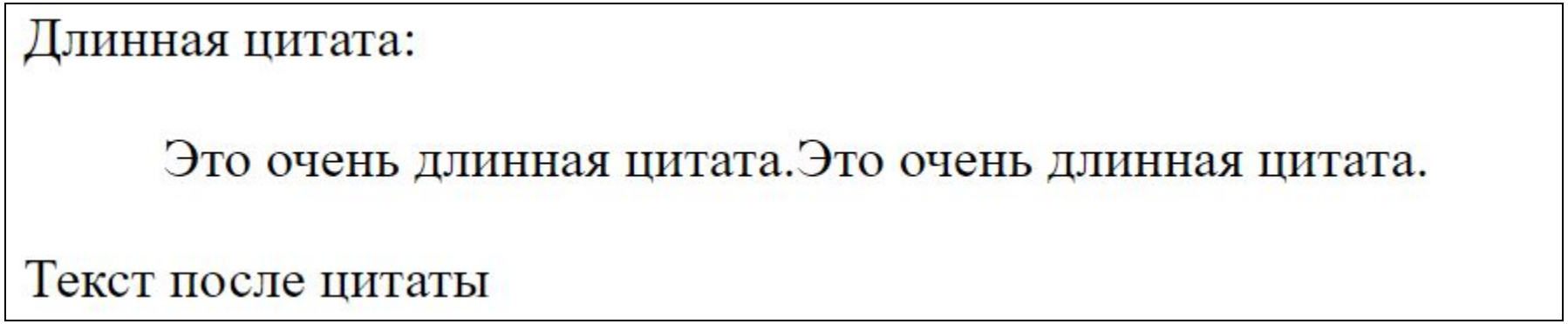
```
Строка №1
    Строка №2
        Строка №3
```

Моноширинный, или непропорциональный шрифт – это шрифт, все знаки которого имеют одинаковую ширину. Этим он отличается от пропорционального шрифта, в котором литеры отличаются по ширине друг от друга

5. Элементы группировки. blockquote

Элемент **blockquote** предназначен для выделения длинных цитат внутри документа. Текст, обозначенный этим тегом, традиционно отображается как выровненный блок с отступами слева и справа, а также с отбивкой сверху и снизу:

```
Длинная цитата:
<blockquote>
   Это очень длинная цитата.Это очень длинная цитата.
</blockquote>
Текст после цитаты
```



6. Заголовки

Элементы **h1**, **h2**, **h3**, **h4**, **h5** и **h6** служат для создания заголовков различного уровня:

```
<h1>Текст</h1>
<h2>Текст</h2>
<h3>Текст</h3>
<h4>Текст</h4>
<h5>Текст</h5>
<h6>Текст</h6>
```

Традиционно элемент h1 определяется только **один раз** и используется для озаглавливания всего документа, h2 – главных его разделов, h3 – подразделов и т.д.

7. Форматирование текста. br. span

Элемент **br** устанавливает перевод строки в том месте, где он находится:

```
Текст с переводом <br /> строки
```

Текст с переводом
строки

Универсальный элемент **span** предназначен для выделения отдельных строк, символов или других строчных элементов для дальнейшего изменения их оформления с помощью CSS. Например, внутри абзаца можно изменить цвет и размер первого слова, если его выделить с помощью элемента span и задать для него желаемый стиль:

```
<span style="color: red;">П</span>ервое слово
```

Первое слово

7. Форматирование текста. small. sup. sub

Элемент **small** уменьшают размер шрифта на единицу по сравнению с обычным текстом:

```
текст <small>текст</small>
```

текст текст

Для отображения верхних и нижних индексов используются элементы **sup** и **sub** соответственно. Например, формула $E_1 = m_1c^2$ выводится таким образом:

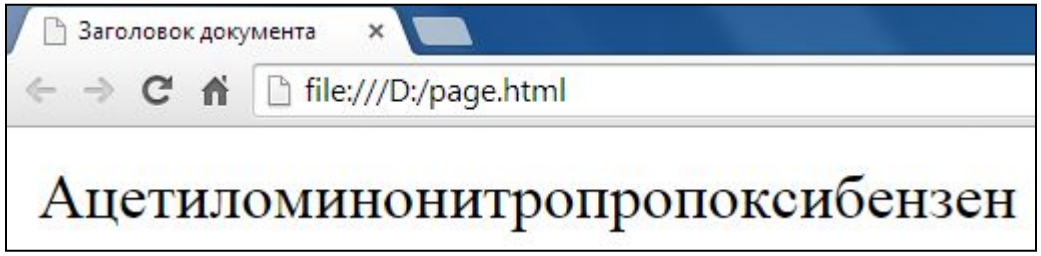
```
E<sub>1</sub> = m<sub>1</sub>c<sup>2</sup>
```

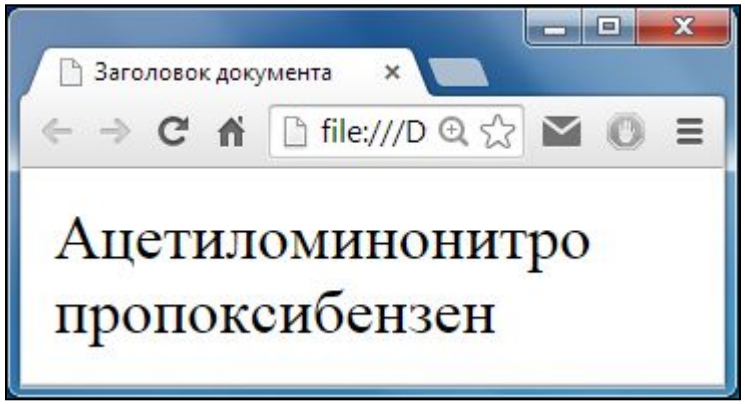
$$E_1 = m_1c^2$$

7. Форматирование текста. wbr

Иногда необходимо расставить в словах так называемые "мягкие" переносы, т.е. обозначить места, в которых допускается переносить слово на следующую строку. Для этого предназначен элемент **wbr**:

```
Ацетило<wbr />мино<wbr />нитро<wbr />пропокси<wbr />бензен
```





7. Форматирование текста. b. strong

Следующие элементы целесообразно применять для **семантического** (логического, смыслового) обозначения ключевых фраз в тексте документа. Несмотря на одинаковый визуальный эффект некоторых из них, необходимо различать их назначение

Элементы **b** и **strong** устанавливают жирное начертание текста:

```
<b>текст</b><br />
<strong>текст</strong>
```

текст
текст

Обратите внимание, что не смотря на схожий результат элементов b и strong, они отличаются между собой тем, что элемент b предназначен для физической разметки текста и устанавливает жирное начертание, а элемент strong служит логической разметкой и определяет важность выделенного текста для поисковых систем

7. Форматирование текста. i. em. kbd

Элементы **i** и **em** нужны для того, чтобы сделать текст курсивом, то есть наклонным:

```
<i>текст</i><br />
<em>текст</em>
```

текст
текст

Обратите внимание, как b и strong, элементы i и em отличаются своими свойствами. Элемент i служит для оформления текста, а em для акцентирования текста для поисковых систем

Элемент **kbd** используется для обозначения текста, который набирается на клавиатуре или для названия клавиш:

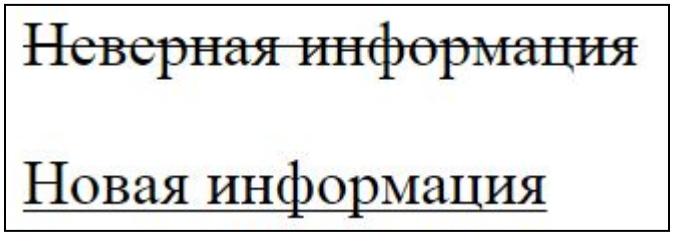
```
Сохранить документ: <kbd>Ctrl+S</kbd>
```

Сохранить документ: Ctrl+S

7. Форматирование текста. del. ins. s. u

Обозначить ошибку в тексте можно элементом **del**, а вставленный текст – элементом **ins**. Несмотря на то, что это строчные теги (не разрывают строки), внутри них допускается размещение блочных элементов:

```
<del>Неверная информация</del>
<ins>Новая информация</ins>
```



Обратите внимание, перечеркнуть текст можно также элементом **s**. Но в отличие от del, он обозначает не ошибку, а потерявшую актуальность или устаревшую информацию. В свою очередь элемент **u** подчеркивает текст, однако, в отличие от ins, назначение его – обратить внимание на некоторую особенность, вроде грамматической ошибки, зачастую вместо элемента u используется более подходящий по логике элемент (например em)

7. Форматирование текста. q. dfn. abbr

Элемент **q** используется для выделения в тексте цитат:

```
Станислав Лец утверждал: <q>Чаще всего выход там, где был вход</q>
```

Станислав Лец утверждал: "Чаще всего выход там, где был вход"

Элемент **dfn** применяется для выделения таких терминов при их первом появлении в тексте:

```
<dfn>Моноширинный</dfn> – это шрифт, все знаки которого имеют одинаковую ширину
```

Моноширинный – это шрифт, все знаки которого имеют одинаковую ширину

Элемент **abbr** указывает, что последовательность символов является аббревиатурой:

```
<abbr title="HyperText Markup Language">HTML</abbr>
```

7. Форматирование текста. cite. time. mark

Элемент **cite** помечает текст как цитату или сноску на другой материал:

```
Сайт: <cite>http://example.com</cite>
```

Сайт: *http://example.com*

Элемент **time** помечает текст внутри как дата, время или оба значения:

```
Дата и время: <time>2014-09-22 18:30</time>
```

Дата и время: 2014-09-22 18:30

Элемент **mark** помечает текст как выделенный. В браузере фоновый цвет текста внутри элемента выделяется жёлтым цветом. Авторы обычно используют mark для привлечения внимания читателя к части текста. Заметьте, что такой текст ничего не говорит о важности выделенного фрагмента, а лишь предлагает обратить на него внимание:

```
текст <mark>текст</mark>
```

текст текст

7. Форматирование текста. code. var. samp

Элемент **code** предназначен для отображения одной или нескольких строк текста, который представляет собой программный код:

```
<code>int a = 5;<br />int b = 10;</code>
```

```
int a = 5;
int b = 10;
```

Элемент **var** используется для выделения переменных компьютерных программ:

```
Переменная <var>name</var> хранит имя пользователя
```

Переменная *name* хранит имя пользователя.

Элемент **samp** используется для отображения текста, который является результатом вывода компьютерной программы или скрипта:

```
Вывод программы: <samp>c = 15</samp>
```

Вывод программы: c = 15

8. Типы элементов. Блочные. Строчные

Каждому элементу присущ свой синтаксис и способ отображения в браузерах. В некоторых запрещено размещать определенное содержимое, другие автоматически переносятся на новую строку и т.д. Несмотря на многообразие элементов, эти правила одинаковы для многих из них, и поэтому логично было бы объединить их в группы по общим свойствам

Одной из таких групп являются **блочные (block)** элементы. Особенность их состоит в том, что они занимают всю ширину родительского элемента, независимо от длины своего содержимого. Таким образом, блочный элемент всегда начинается с новой строки, равно как и следующий за ним. Примерами блочных элементов являются div, p, h1-h6 и pre

В отличие от них элементы, не прерывающие течения строки, называются **строчными (inline)**. Несмотря на то, что br переносит текст на следующую строку, его относят именно к строчным, поскольку он не занимает всю ширину родительского элемента. Примерами строчных элементов являются b, dfn, var и em

8. Типы элементов. Структурные

Строчные элементы могут располагаться как внутри блочных, так и друг в друге. Но блочные не могут применяться внутри строчных, и чаще всего вложение их друг в друга также недопустимо:

Неправильно: `<p><h1>Такое вложение недопустимо</h1></p>`

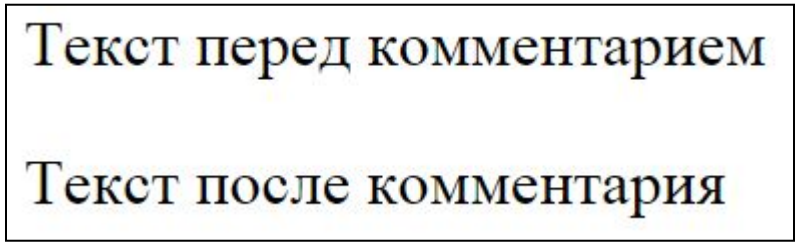
Правильно: `<p>Абзац содержит <b>вложенные <i>строчные</i> теги</b></p>`

Элементы, формирующие таблицы, списки и другие составные конструкции, называются **структурными (structural)**. По одиночке они не применяются, а правила их использования существенно отличаются друг от друга. Примерами структурных элементов являются html, body и head

8. Типы элементов. Комментарии

Тег комментария не похож на остальные и выглядит следующим образом:

```
<p>Текс перед комментарием</p>
<!-- Комментарий между абзацами -->
<p>Текст после комментария</p>
```



Этот тег не поддерживает атрибуты. Применяется он для комментирования исходного кода, чтобы облегчить в дальнейшем его редактирование. Комментарии игнорируются браузером, и не видны пользователю. Однако некоторые программы для работы с HTML-кодом сохраняют в них вспомогательную информацию

Источники

http://webref.ru/
http://professorweb.ru/

Спасибо за внимание!