

Язык гипертекстовой разметки

HTML (Hypertext Markup Language) — язык разметки гипертекста. Под *гипертекстом* понимается текст, включающий дополнительные элементы, в том числе ссылки.

Теги

Теги — единственная синтаксическая конструкция.

а) Виды тегов.

Дескрипторы текста:

<имя параметр=значение ...>	текст	</имя>
<i>открывающий</i>	<i>опции (параметры)</i>	<i>закрывающий</i>

Если значение содержит пробелы или особые символы, то его необходимо заключать в кавычки. Кавычки каждого вида можно использовать внутри строки, заключенной в кавычки другого вида.

Параметры тегов могут опускаться.

Esc—последовательности:

&имя; — общий вид.

Примеры:

< есть **<**

> есть **>**

& есть **&**.

** ** — (non-breaking space) неразрывающий пробел.

Комментарии:

<!-- комментарий --> — содержимое такого тега игнорируется.

б) Правила расположения тегов.

Допускается вложенность тегов:

<тег1> ... <тег2> ... </тег2> ... </тег1>.

Пересечение тегов:

<тег1> ... <тег2> ... </тег1> ... </тег2>

является нарушением синтаксиса и может
интерпретироваться некорректно.

Общая структура документа

<HTML>

<HEAD>

Заголовок документа. При просмотре не отображается.

</HEAD>

<BODY>

Тело — отображаемая часть документа.

</BODY>

</HTML>

Примеры содержимого заголовка

Тег `<TITLE>` **Название документа** `</TITLE>` —
отображается в частности в заголовке окна просмотра.

META-инструкции (содержат служебную информацию):

```
<META NAME="description"  
CONTENT="Аннотация">
```

— определяет переменную `description`, содержащую
краткое описание документа, ориентирована на помощь в
поиске, однако обычно поисковые машины ее игнорируют;

```
<META HTTP-EQUIV="Content-type"  
CONTENT="text/html; charset=windows-1251">
```

— дает браузеру указание интерпретировать загружаемый
документ как содержащий HTML-текст в кодировке
Windows-1251.

Параметры <BODY>

Примеры атрибутов:

bgcolor — цвет фона документа;

text — цвет текста документа;

link — цвет элемента текста, задающего
гипертекстовую ссылку;

vlink — определяет цвет ссылки на документ, который
уже был просмотрен ранее;

alink — определяет цвет ссылки в момент, когда на нее
указывает курсор мыши и нажата ее кнопка.

background="имя файла" — изображение, служащее
фоном. Допускаются формат GIF и JPEG.

Пример: **bgcolor=#FFFFFF** — белый цвет фона.

Форматирование текста

а) Заголовки.

Тег заголовка имеет вид: **<Hn> Текст заголовка </Hn>**, где *n* – число, задающее размер заголовка от 1 (самый крупный) до 6 (самый мелкий). Тег может содержать некоторые параметры форматирования, присущие абзацу.

б) Форматирование абзацев.

Абзац задается тегом

<P> содержимое абзаца </P>.

Параметр **ALIGN** устанавливает выравнивание абзаца. Допустимые значения: **ALIGN=LEFT** (выравнивание влево), **ALIGN=CENTER** (выравнивание по центру), **ALIGN=RIGHT** (выравнивание вправо).

Браузеры (программы просмотра html–документов), также как и, например, компиляторы, воспринимают любое количество пробелов, табуляторов и переходов на новую строку как один пробел. Поэтому для форматирования текста нужны специальные средства.

Тег `
` — принудительный переход на новую строку (непарный тег).

Тег `<PRE> ... </PRE>` — текст, заключенный между этими метками (от английского `preformatted` — предварительно форматированный), выводится браузером на экран как есть — со всеми пробелами, символами табуляции и конца строки. Это удобно использовать, например, при включении фрагментов программного кода.

в) Списки.

Текст, расположенный между метками `` и ``, воспринимается как нумерованный список. Каждый новый элемент списка следует начинать с метки `` (непарный тег). Например,

```
<UL>
```

```
<LI> Иванов;
```

```
<LI> Петров;
```

```
<LI> Сидоров.
```

```
</UL>
```

Нумерованные списки: `` ... `` устроены точно так же, как нумерованные, только вместо символов, выделяющих новый элемент, используются числа.

Шрифты

а) Логические стили.

` ... ` — акцент (emphasis),

` ... ` — сильный акцент,

— предписывают выделить заключенный между ними текст, не конкретизируя как именно.

б) Физические стили.

` ... ` — жирный шрифт,

`<I> ... </I>` — наклонный шрифт,

`<TT> ... </TT>` — имитирующий пишущую машинку.

Тег ` ... ` — спецификация шрифта.

Параметры: `SIZE=число` — размер шрифта;

`COLOR=#rrggbb` — цвет.

Ссылки и переходы

а) Формат ссылки.

** выделенный фрагмент текста **

б) Относительная и абсолютная адресация.

Если адрес перехода начинается не с косой черты, переход будет выполнен относительно текущего каталога. Например:

** Перейти к оглавлению **

Такая адресация называется *относительной*.

Для перехода в родительский каталог в пути указывается «../».

Если в адресе перехода не указан сервер, переход будет выполнен на текущем сервере.

Если необходимо дать ссылку на документ, находящийся на другом сервере, то используется двойная косая черта, за которой указывается имя компьютера, например:

```
<A  
  HREF="http://yi.com/home/ChuvakhinNikolai/pr.htm">  
  Практическое руководство по HTML </A>.
```

в) Анкеры.

При необходимости можно задать переход к определенному месту внутри этого документа.

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

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

Заметим, что «#» является корректной ссылкой и означает переход в текущую позицию документа.

Таблицы

а) Структура.

```
<TABLE BORDER=1> <!-- начало таблицы-->
  <CAPTION Заголовок </CAPTION>
  <TR>                                <!-- начало первой строки-->
    <TD>                                <!-- начало первой ячейки-->
      Первая строка, первая колонка
    </TD>                                <!-- конец первой ячейки-->
    <TD>    Первая строка, вторая колонка </TD>
  </TR>
  <TR>
    <TD>
      Вторая строка, первая колонка
    </TD>
    <TD>
      Вторая строка, вторая колонка
    </TD>
  </TR>
</TABLE>
```

б) Тег **<TABLE>**.

Таблица содержится в теге **<TABLE>**, который может включать несколько атрибутов:

ALIGN — устанавливает расположение таблицы по отношению к полям документа. Допустимые значения: **LEFT**, **CENTER**, **RIGHT**.

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

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

CELLSPACING — устанавливает расстояние между рамками ячеек таблицы в пикселях (например, **CELLSPACING=2**).

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

Таблица может иметь заголовок (**<CAPTION> ... </CAPTION>**).

в) Строки.

Каждая строка таблицы задается парным тегом **<TR>**, который может включать следующие атрибуты:

ALIGN — устанавливает выравнивание текста в ячейках строки.

VALIGN — устанавливает вертикальное выравнивание текста в ячейках строки. Допустимые значения: **top**

(выравнивание по верхнему краю), **MIDDLE** (выравнивание по центру), **BOTTOM** (выравнивание по нижнему краю).

г) Ячейки.

Каждая ячейка таблицы парным тегом **<td>**, который может включать следующие атрибуты:

nowrap — Присутствие этого атрибута означает, что содержимое ячейки должно быть показано в одну строку;

colspan — устанавливает "размах" ячейки по горизонтали, например, **colspan=3** означает, что ячейка простирается на три колонки;

rowspan — устанавливает "размах" ячейки по вертикали, например, **rowspan=2** означает, что ячейка занимает две строки;

ALIGN, VALIGN — аналогичны строке;

WIDTH — устанавливает ширину ячейки в пикселях;

HEIGHT — устанавливает высоту ячейки в пикселях.

д) Замечания.

Если ячейка таблицы пуста, вокруг нее не рисуется рамка. Если ячейка пуста, а рамка нужна, можно внести невидимый символьный объект ** sp;** (обычный пробел будет проигнорирован браузером).

Заметим, что любая ячейка таблицы может содержать в себе другую таблицу.

Таблицы в HTML используются очень широко, причем не только и не столько для создания таблиц в привычном понимании, сколько как средство структуризации текста на экране.

Встраиваемые объекты

а) Рисунки.

Стандарт HTML допускает вставку в текст ссылок на графические файлы формата JPEG и GIF, причем последние могут быть анимированными.

Тег вставки рисунка:

```

```

Здесь **src** — URL-адрес файла с изображением.

б) Линии.

Линии (закрашенные бары) вставляются тегом

```
<HR size=2 width=50%>
```

Размер объектов обычно задается в условных единицах или в процентах от размера окна.

Формы

Формы предназначены для ввода информации.

а) Структура.

Форма задается тегом

<FORM ACTION=...> Содержимое формы. </FORM>

б) Поля.

<INPUT TYPE=тип NAME=идентификатор VALUE=значение>

Типы полей:

text — текстовое поле;

radio — переключатель;

button — кнопка;

submit — кнопка отсылки формы.

Идентификатор используется для ссылок на поля.

в) Пример.

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

```
<H2 align=center> Регистрационная форма. </H2>
```

```
<P align=right>
```

```
Имя: <INPUT TYPE=text SIZE=40 NAME=fn><BR>
```

```
Фамилия: <INPUT TYPE=text SIZE=40 NAME=ln><BR>
```

```
Пол: <INPUT TYPE=radio NAME=gender
      VALUE="male" checked> мужской
```

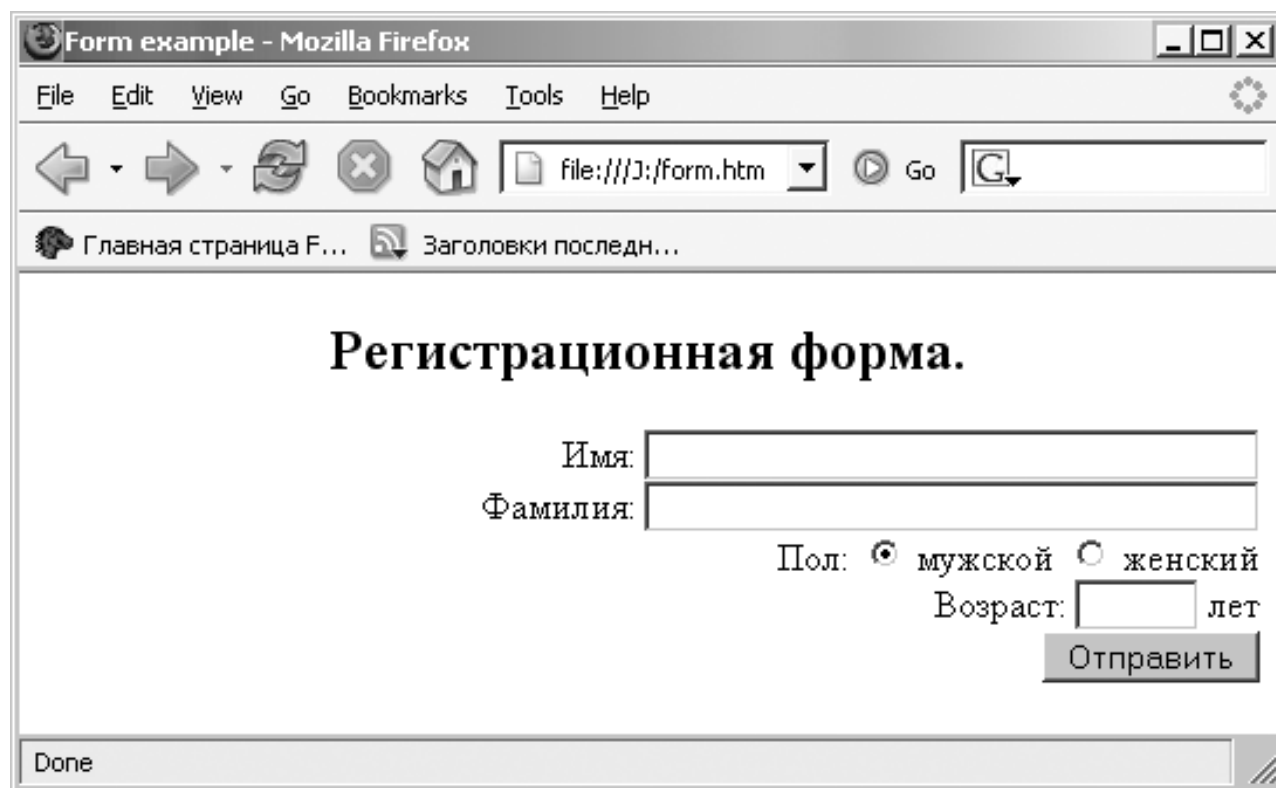
```
<INPUT TYPE=radio NAME=gender
      VALUE="female"> женский<BR>
```

```
Возраст: <INPUT TYPE=text SIZE=5 NAME=age>
          лет<BR>
```

```
<INPUT TYPE=submit VALUE="Отправить">
```

```
</P> </FORM>
```

При вставке в тело документа данная форма будет выглядеть следующим образом.



The screenshot shows a Mozilla Firefox browser window titled "Form example - Mozilla Firefox". The address bar displays "file:///J:/form.htm". The browser's menu bar includes File, Edit, View, Go, Bookmarks, Tools, and Help. The toolbar contains navigation buttons (back, forward, stop, home) and a search bar. The main content area displays the registration form titled "Регистрационная форма.".

Регистрационная форма.

Имя:

Фамилия:

Пол: ☒ мужской ☐ женский

Возраст: лет

Done

Предположим теперь, что пользователь указал: имя Иван, фамилию Петров, мужской пол и возраст 22 года. Теперь если он нажмет на кнопку «Отправить», то на адрес **user@mail.box** электронной почтой будет отправлено сообщение следующего содержания:

fn=Иван

ln=Петров

gender=male

age=22

Отправка электронной почты является не основным видом действия (**action**). Более распространен вариант отправки данных на сервер для их автоматической обработки.

JavaScript

Язык JavaScript является интерпретируемым языком, инструкции которого выполняются браузером при просмотре документа.

Базовые конструкции языка

а) Объекты.

Обращение к объектам и их свойствам:

объект.свойство

Для обращения доступны predefined объекты:

document — инкапсулирует содержимое текущего документа;

location — содержит информацию об URL;

window — инкапсулирует параметры окна просмотра.

б) Функции.

function имя(параметры) {операторы}

в) Переменные.

var имя;

Включение элементов JavaScript в документ

а) Тег кода.

```
<script language="JavaScript">  
...код...  
</script>
```

б) Использование в ссылке.

```
<a href="javascript:...код...">
```

Код, помещенный в ссылку, будет исполняться вместо перехода при активации ссылки.

в) Обработчик события.

Код обработки событий помещается в значение соответствующих параметров тегов, допускающих обработку событий. Примеры таких параметров: `onMouseOver`, `onClick` и т. п.

Примеры

а) Работа с формой.

```
<html>  <head>
<title>Form script demo</title>
<script language="JavaScript">
function f1() {
    alert("The value of the textelement is: " +
        document.myForm.myText.value);
} </script> </head>
<body>
<form name="myForm">
<input type="text" name="myText" value="change">
<input type="button" value="Press" onClick="f1()">
</form> </body> </html>
```

б) Использование таймера.

Функция `setTimeout` выполняет указанную команду через заданное число миллисекунд.

```
<html>
<head>
<script language="JavaScript">
function timer() {
    setTimeout("alert('Прошло 3 секунды')", 3000);
}
</script>
<body OnLoad="timer()">
    ...
</body>
</html>
```

в) Строка состояния.

Следующий фрагмент иллюстрирует выдачу сообщения в строку состояния.

```
<a href="file.htm"  
  onMouseOver="window.status='текст'; return  
true;"  
  onMouseOut="window.status='';"> ... </a>
```

После вывода текста метод возвращает `true`, чтобы следом не был запущен стандартный обработчик события, который перезаписывает строку состояния.

г) Изменение изображения.

Следующий пример демонстрирует возможность изменения изображения внутри ссылки при наведении мыши.

```
<a href="#"  
  onMouseOver="document.myImage.src='img2.gif'"  
  onMouseOut="document.myImage.src='img1.gif'">  
 </a>
```

Можно усовершенствовать приведенный код, реализовав предварительную загрузку изображения в скрытый объект, что позволит исключить возможные задержки с отрисовкой.

Каскадные таблицы стилей

Каскадные таблицы стилей (CSS — Cascade Style Sheets, что дословно переводится как каскадные листы стилей) предназначены для решения двух задач: дать разработчику (дизайнеру) полный контроль над представлением документа в браузере и отделить содержание документа от его представления. Одновременно достигается сокращение дублирования в задании параметров.

Способы изменения стилей

а) Переопределение в элементе.

Стиль можно переопределить в самом элементе с помощью параметра **style**.

```
<h1 style="font-size:10pt;">
```

Список свойств, переопределяемых посредством стилей, очень обширен и может быть найден в справочной литературе.

б) Специальный тег.

```
<STYLE> p {color:darkred;text-align:justify;  
           font-size:8pt;} </STYLE>
```

Такое переопределение следует делать в заголовке документа.

в) Ссылка на внешнее описание.

Более удобно определять стили в отдельном файле, на который впоследствии можно ссылаться из различных документов.

```
<LINK TYPE="text/css" REL="stylesheet"  
      HREF="http://mysite.ru/my_style.css">
```

Тег имеет широкое назначение, поэтому при ссылке на стили параметры **TYPE** и **REL** обязательны и должны иметь значения как указано в примере.

г) Импорт.

Альтернативой ссылке является механизм импорта в теге **<STYLE>**, причем строка импорта должна быть в нем первой.

```
<STYLE> @import:url(http://mysite.ru/style.css)  
</STYLE>
```

Конструирование стилей

Язык CSS позволяет задавать так называемые классы стилей, которые можно считать аналогом макроопределений в языках программирования (в частности, макросов **#define** в C).

а) Синтаксис.

Определение стиля выглядит следующим образом.

селектор, ... {атрибут: значение;}

Если селекторы перечислены через ‘,’ , это эквивалентно отдельным определениям для каждого селектора.

Если селекторы перечислены через пробел, то вводимый стиль применяется только к элементам, которые находятся в тегах, вложенных по отношению друг к другу.

б) Селектор.

Селектор имеет формат

тег.класс

При этом как **тег**, так и **.класс** могут опускаться (во втором случае точка также не ставится).

Здесь **класс** обозначает вводимый уникальный идентификатор определяемого класса.

Если **тег** опущен, то введенный класс может использоваться в любом теге, иначе только для указанного.

В качестве селектора можно указать также идентификатор (значение параметра **id**) объекта

#идентификатор

в) Использование.

<h1 class = "...">

Блочные и строковые элементы

Возможность изменения стиля делает непринципиальными различия в больших группах тегов. Можно выделить две большие группы эквивалентных в этом смысле тегов: *блочные* и *строковые* элементы.

К блочным относятся элементы, структурирующие текст, например тег **<P>**. Строковые элементы определяют внешний вид фрагмента текста, пример — тег ****.

Возможность переопределения стиля приводит к тому, что элементы каждой из этих групп взаимозаменяемы между собой, поэтому введены специальные теги для их «универсальных» представителей:

<DIV> — универсальный блочный элемент;

**** — универсальный строковый элемент.