

Тема 2.2

«Использование HTML для создания веб-страниц»

Вступление

Итак, вёрстка сайта. В данной теме мы рассмотрим основные особенности HTML и CSS, которые могут нам понадобиться при разработке веб-ориентированных приложений вообще и веб-сайтов в частности.

Также следует подчеркнуть, что более полную информацию по данным вопросам можно получить в курсе «Веб-дизайн», мы же рассмотрим только основные моменты.

Притупим.

HTML: общие сведения

HTML (Hyper Text Markup Language – язык гипертекстовой разметки) – это стандартный язык разметки документов в Интернет. Веб-страницы создаются при помощи языка HTML (или XHTML).

Язык HTML интерпретируется браузером и отображается в виде документа, в удобной для человека форме. HTML является подмножеством SGML (Standard Generalised Markup Language – стандартный обобщённый язык разметки) и соответствует международному стандарту ISO 8879.

Язык HTML был разработан британским учёным Тимом Бернерсом-Ли приблизительно в 1991-1992 годах в стенах Европейского совета по ядерным исследованиям в Женеве.

HTML создавался как язык для обмена научной и технической документацией, пригодный для использования людьми, не являющимися специалистами в области вёрстки.

HTML: общие сведения

HTML успешно справлялся с проблемой сложности SGML путём определения небольшого набора структурных и семантических элементов (размечаемых «тегами»), служащих для создания относительно простых, но красиво оформленных документов.

Помимо упрощения структуры документа, в HTML внесена поддержка гипертекста. Мультимедийные возможности были добавлены позже. Изначально язык HTML был задуман и создан как средство структурирования и форматирования документов без их привязки к средствам воспроизведения (отображения).

В идеале, текст с разметкой HTML должен был без стилистических и структурных искажений воспроизводиться на оборудовании с различной технической оснащённостью. Однако современное применение HTML очень далеко от его изначальной задачи.

С течением времени, основная идея платформонезависимости языка HTML была отдана в своеобразную жертву современным потребностям в мультимедийном и графическом оформлении.

HTML: история

Текстовые документы, содержащие код на языке HTML в основном отображаются веб-браузерами. Наиболее популярными на сегодняшний день браузерами являются Internet Explorer, Firefox, Safari, Google Chrome и Opera.

Версии HTML

- RFC 1866 - HTML 2.0, одобренный как стандарт 22 сентября 1995;
- HTML 3.2 - 14 января 1997;
- HTML 4.0 - 18 декабря 1997; HTML 4.01 (изменения, причём более значительные, чем кажется на первый взгляд) - 24 декабря 1999;
- ISO/IEC 15445:2000 (так называемый ISO HTML, основан на HTML 4.01 Strict) - 15 мая 2000;
- HTML 5 - в разработке.
- Официальной спецификации HTML 1.0 не существует.

HTML: история

До 1995 года существовало множество неофициальных стандартов HTML. Чтобы стандартная версия отличалась от них, ей сразу присвоили второй номер.

Версия 3 была предложена Консорциумом Всемирной паутины (World Wide Web Consortium, W3C) в марте 1995 года и обеспечивала много новых возможностей, таких как создание таблиц, «обтекание» изображений текстом и отображение сложных математических формул.

Даже при том, что этот стандарт был совместим со второй версией, реализация его была сложна для браузеров того времени.

Версия 3.1 официально никогда не предлагалась, и следующей версией стандарта HTML стала 3.2, в которой были опущены многие нововведения версии 3.0, но добавлены нестандартные элементы, поддерживаемые браузерами «Netscape» и «Mosaic».

HTML: история

HTML версии 4.0 содержит много элементов, специфичных для отдельных браузеров, но в то же время произошла некоторая «очистка» стандарта. Многие элементы были отмечены как устаревшие и нерекомендованные (**англ. «deprecated»**).

В частности, элемент `font`, используемый для изменения свойств шрифта, был помечен как устаревший (вместо него рекомендуется использовать таблицы стилей CSS).

Начиная с 2004 года, сообществом WHATWG (Web Hypertext Application Technology Working Group), ведётся разработка HTML версии 5.

Сейчас Консорциумом Всемирной паутины (W3C) разрабатывает пятую версию языка HTML5. Черновой вариант спецификации языка появился в Интернете 20 ноября 2007.

HTML: история

Параллельно ведётся работа по дальнейшему развитию HTML под названием XHTML (Extensible Hypertext Markup Language – расширяемый язык гипертекстовой разметки). Пока XHTML по своим возможностям сопоставим с HTML, однако предъявляет более строгие требования к синтаксису.

Как и HTML, XHTML является подмножеством языка SGML, однако XHTML, в отличие от предшественника, основан на XML.

Вариант XHTML 1.0 был одобрен в качестве Рекомендации Консорциума Всемирной паутины (W3C) 26 января 2000 года. Планируемая спецификация XHTML 2.0 разрывает совместимость со старыми версиями HTML и XHTML, что не очень устраивает некоторых веб-разработчиков и производителей браузеров.

Группой WHATWG (Web Hypertext Application Technology Working Group) разрабатывается спецификация Web Applications 1.0, часто неофициально называемая <HTML5>, которая расширяет HTML (впрочем, имея и совместимый с XHTML 1.0 XML-синтаксис) для лучшего представления семантики различных типичных страниц, например форумов, сайтов аукционов, поисковых систем, онлайн-магазинов и т. д., которые не очень удачно вписываются в модель XHTML 2.

HTML: проблемы совместимости

В середине 1990-х годов возникло следующее явление: основные производители браузеров – компании Netscape и Microsoft – начали внедрять собственные наборы элементов в HTML-разметку.

Создалась путаница из различных конструкций, доступных для просмотра то в одном, то в другом браузере. Особенно большие трудности были при создании кросс-браузерных программ на языке JavaScript.

Веб-мастерам приходилось создавать несколько вариантов страниц или прибегать к другим ухищрениям. На какое-то время проблема потеряла актуальность по двум причинам:

- из-за вытеснения браузером Microsoft Internet Explorer всех остальных браузеров;

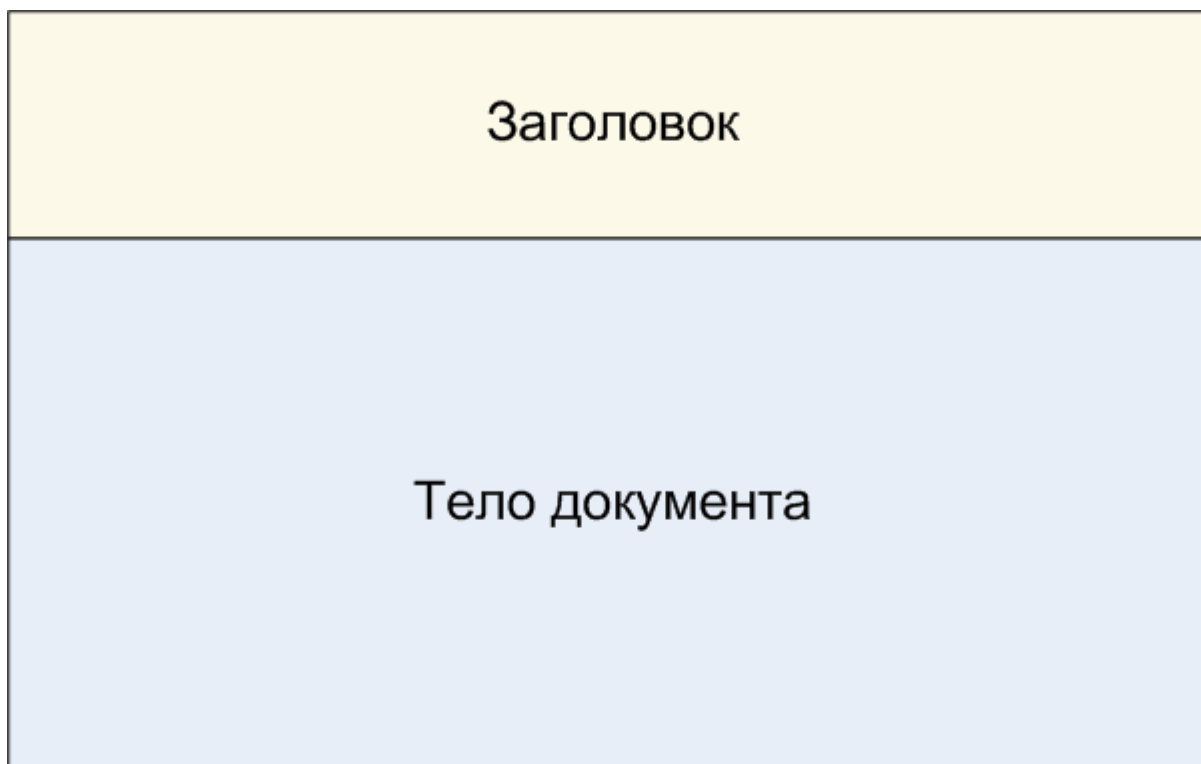
- благодаря усилиям производителей других браузеров, которые либо следовали стандартам W3C (как Mozilla и Opera), либо пытались создать максимальную совместимость с Internet Explorer.

На современном этапе можно констатировать рост популярности браузеров, следующих рекомендациям W3C (это Mozilla Firefox и другие браузеры на движке Gecko; Konqueror, Safari и другие браузеры на движке KHTML; Opera с уникальным движком Presto). При этом Internet Explorer пока сохраняет лидирующие позиции.

HTML: структура документа

Обобщённо HTML-документ состоит из заголовка и тела, внутри которых расположены теги и человекополезная информация.

Примечание: HTML – НЕрегистрочувствительный язык.



HTML: теги

Тег – специальная конструкция, заключённая в <угловые скобки>, предназначенная для браузера или иного ПО, выполняющего анализ («парсинг») HTML-документа.

Теги бывают парными: `<p>тут какой-то текст</p>`
и непарными: `
`

У тегов могут быть атрибуты: `<p align="left">какой-то текст</p>`

ВАЖНО! Значения атрибутов следует **ОБЯЗАТЕЛЬНО** брать в кавычки! Во-первых, этого требует стандарт XHTML, во-вторых, представим себе ситуацию, когда в процессе генерации формы значением поля оказываются два слова: «Василий Пупкин». Код примет вид (пример НЕПРАВИЛЬНОГО КОДА!):

```
<input type=text value=Василий Пупкин />
```

Браузер воспримет как значение поля только слово «Василий». А слово «Пупкин» отбросит как нераспознанную последовательность символов.

HTML: теги, атрибуты

Последовательность атрибутов не имеет значения.

Атрибуты, указывающие размеры элементов могут указываться в процентах от размеров родительского элемента (`width="90%"`) или в пикселах (`height="20"`).

Атрибуты, указывающие цвет, могут записываться в виде слов (`color="red"`) или в виде указания RGB-записи цвета (`color="#FF0000"`).

Подробнее об атрибутах того или иного тега мы поговорим в соответствующих разделах, посвящённых конкретным тегам. Учитывая количество и разнообразие атрибутов, полную информацию по ним рекомендуется смотреть в спецификации HTML.

HTML: структура

HTML-документ – это один большой контейнер, который начинается с тега `<HTML>` и заканчивается тегом `</HTML>`.

Контейнер HTML или гипертекстовый документ состоит из двух других вложенных контейнеров: заголовка документа (HEAD) и тела документа (BODY).

Рассмотрим простейший пример классического документа:

```
<HTML>
  <HEAD>
    <TITLE>Простейший документ</TITLE>
  </HEAD>
  <BODY TEXT="#0000ff" BGCOLOR="#f0f0f0">
    Какой-то текст
  </BODY>
</HTML>
```

HTML: структура

Компания Netscape Communication расширила классическую форму документа возможностью организации фреймов (кадров), позволяющих разделить рабочее окно программы просмотра на несколько независимых фреймов.

В каждый фрейм можно загрузить свою страницу HTML.

Пример документа с фреймами:

```
<HTML>
  <HEAD>
    <TITLE>Документ с фреймами</TITLE>
  </HEAD>
  <FRAMESET COLS="30%,*">
    <FRAME SRC="frame1.html" NAME=LEFT>
    <FRAME SRC="frame2.html" NAME=RIGHT>  </FRAMESET>
</HTML>
```

В настоящий момент фреймы практически не используются по следующим причинам:

- страницы с фреймами плохо индексируются поисковыми системами;
- на страницу с фреймами невозможно дать ссылку, клик по которой привёл бы к отображению того же вида документа, который наблюдал тот, кто дал ссылку;
- появление технологии AJAX позволило достичь всех преимуществ фреймов практически избавившись от их недостатков.

HTML: заголовок

Первоначально существование заголовка определялось необходимостью именования окна браузера. Это достигалось за счет тега TITLE:

```
<HTML>  
  <HEAD>  
    <TITLE>Это заголовок</TITLE>  
  </HEAD>  
  <BODY>  
    ...  
  </BODY>  
</HTML>
```

Однако задумывался заголовок для несколько иных целей.

HTML: заголовок

Исходя из общих соображений работы гипертекстовых систем, все гипертекстовые связи принято разделять на контекстные и общие.

Контекстные связи соответствуют определённому месту документа – контексту. В HTML такие связи реализованы в виде гипертекстовых ссылок (тег A (anchor)).

Фактически до реализации CSS в современных браузерах это был единственный вид связей, которыми мог управлять автор HTML-документа.

Общие гипертекстовые связи определяются не частью документа (контекстом), а всем документом целиком. Например, быть предыдущим по отношению к другому документу или следующим – это общая гипертекстовая связь, которая позволяет организовать так называемый "линейный" просмотр информационных узлов гипертекстовой сети.

Реализация такого сорта ссылок уже давно является частью проектов W3C (Arena, Amaya).

HTML: заголовок

В коммерческих браузерах такой механизм реализован только для описателей стилей (тег LINK).

Важную роль заголовков HTML-документа играет в JavaScript. Существует принципиальная разница между заголовком и телом документа при использовании тега SCRIPT. Она заключается в определении зоны видимости функций и переменных.

Переменные и функции, определённые в заголовке документа, относятся ко всему окну браузера. Это значит, что к ним можно обратиться из любого места документа и изменить их значения.

Кроме того, к ним можно обратиться из другого окна или фрейма. Фактически, это глобальные переменные.

HTML: заголовок

Ещё одной функцией заголовка HTML-документа является управление HTTP-обменом через тег META.

Следует упомянуть ещё об одном важном назначении заголовка HTML-документа – поисковом образе документа для индексирования роботами поисковых систем.

Тег META позволяет хранить списки ключевых слов и описание документа, которые будут использоваться для составления индекса поисковой системы и появляться в качестве описания документа в случае выдачи ссылки на него при поиске по ключевым словам.

Хотя многие поисковые системы в настоящее время используют свои алгоритмы анализа HTML-документов, такой подход по-прежнему считается хорошим тоном и часто применяется.

HTML: заголовок, основные элементы

Основные теги, встречающиеся внутри заголовка HTML-документа – это:

- TITLE (заглавие документа);
- META (метаинформация);
- LINK (общие ссылки);
- STYLE (описание стилей);
- SCRIPT (скрипты).
- BASE (базовый URL);
- ISINDEX (поисковый шаблон).

Чаще всего применяются теги TITLE, SCRIPT, STYLE. BASE и ISINDEX в последнее время практически не применяются. LINK указывают только при использовании внешних относительно данного документа описаний стилей.

HTML: заголовок, тег TITLE

Тег TITLE служит для именования документа. Более прозаическое его назначение – именование окна браузера, в котором просматривается документ.

В различных браузерах алгоритм отображения элемента TITLE может отличаться.

При выборе текста для содержания TITLE следует учитывать, что отображается он системным шрифтом, так как является заголовком окна браузера.

В нелокализованных версиях операционных систем русский текст содержания элемента TITLE будет отображаться абракадаброй.

Синтаксис TITLE в общем виде выглядит следующим образом:

`<TITLE>название документа</TITLE>`

Роботы многих поисковых систем используют содержание элемента TITLE для создания поискового образа документа.

HTML: заголовок, тег META

META – наиболее популярный тег, более распространён только тег TITLE.

META содержит управляющую информацию, которую браузер использует для правильного отображения и обработки содержания тела документа.

Впервые тег META был задействован при принудительной перезагрузке документа браузером через заголовок HTTP-сообщения. В заголовке HTTP-сообщения можно указать:

```
<META HTTP-EQUIV="Refresh" CONTENT="1;  
URL=refresh.htm">
```

В данном случае через одну секунду после загрузки документа браузер должен инициировать загрузку страницы refresh.htm.

HTML: заголовок, тег META

Также с помощью META можно указать тип кодировки документа – CHARSET:

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

Также с помощью META можно попытаться запретить кэширование документа (клиентское ПО может проигнорировать запрет):

```
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
```

Параметр Pragma – это наследие HTTP 1.0.

В новой версии протокола HTTP (HTTP 1.1) управление кэшированием осуществляется через параметр Cache-Control:

```
<META HTTP-EQUIV="Cache-Control"          CONTENT="no-  
cache">
```

Новый механизм управления кэшированием и хранением документа на стороне клиента гораздо более гибок, чем в HTTP 1.0. Например, можно запретить хранение документа после пересылки:

```
<META HTTP-EQUIV="Cache-Control" CONTENT="no-store">
```

Точно так же можно задать время последней модификации (Last-Modified) или дату истечения актуальности документа (Expire).

HTML: заголовок, тег META

С появлением роботов поисковых машин на тег META была возложена еще одна функция – описание поискового образа документа. Наиболее последовательно это было впервые реализовано в Webcrawler.

До этого в качестве поискового образа документа использовался либо весь список слов документа, либо слова первого абзаца.

Собственно, для описания документа используется два тега META. Один определяет список ключевых слов, а второй – «реферат» документа, который отображается в качестве пояснения к ссылке на документ в отчёте поисковой машины о выполненном запросе.

```
<META NAME="description" http-equiv="description"  
content="Изучение HTML. Теги заголовка.">
```

```
<META NAME="keywords" HTTP-EQUIV="keywords"  
CONTENT=«учебное пособие; web-технологии; web;  
технология; HTML">
```

В общем случае тег META выглядит следующим образом:

```
<META [name=имя] [HTTP-EQUIV=имя_HTTP-оператора]  
CONTENT=текст>
```

HTML: заголовок, тег LINK

Тег LINK позволяет загружать внешние описания стилей:

```
<LINK          REL="stylesheet"          href="../css/style.css"  
TYPE="text/css">
```

Атрибут REL определяет тип гипертекстовой связи, HREF (Hypertext REFerence) указывает адрес документа, идентифицирующего связь, а атрибут TYPE определяет тип содержания этого документа.

В общем случае контейнер LINK имеет следующий вид:

```
<LINK          [REL=тип_отношения]          [HREF=URL]  
[TYPE=тип_содержания]>
```


HTML: заголовок, тег STYLE

STYLE предназначен для размещения описаний стилей. При этом описание стиля из данного тега, если оно совпадает с именем класса и/или идентификатором подкласса, описанным во внешнем файле, заменяет описание стиля из внешнего файла.

С точки зрения влияния на весь документ, описания стилей задают правила отображения контейнеров HTML-документа для всей страницы.

Пример:

```
<STYLE>  
.redfont {color: red}  
</STYLE>
```

В общем виде запись STYLE выглядит так:

```
<STYLE> описание стиля/стилей </STYLE>
```

HTML: заголовок, тег SCRIPT

SCRIPT служит для размещения кода JavaScript или VBScript.

Вообще говоря, SCRIPT можно использовать не только в заголовке документа, но и в его теле. В отличие от STYLE, ему не требуется дополнительный контейнер LINK для загрузки внешних файлов кодов. Это можно сделать непосредственно в самом SCRIPT:

```
<SCRIPT LANGUAGE="JavaScript" SRC="script_file" />
```

В общем виде запись контейнера выглядит следующим образом:

```
<SCRIPT [TYPE=тип_языка_программирования]  
[SRC=URL]> JavaScript/VBScript-код </SCRIPT>
```

Существует несколько скриптовых языков: JavaScript, VBScript, JScript (фактически, альтернативная реализация JavaScript). По умолчанию подразумевается JavaScript.

HTML: заголовок, теги BASE и ISINDEX

BASE служит для определения базового URL для гипертекстовых ссылок документа, заданных в неполной (частичной) форме. Кроме того, BASE позволяет определить мишень (окно) (target) загрузки документа по умолчанию при выборе гипертекстовой ссылки текущего документа.

Применение BASE в современных документах ограничено.

ISINDEX используется для указания поискового шаблона и унаследован от ранних версий HTML. В HTML 4.0 этот тег не определён.

HTML: заголовок, иконка сайта

С помощью тега LINK также можно устанавливать «иконку сайта» — изображение, которое будет отображаться в браузере и в некоторых поисковых системах.

```
<link rel="icon" href="/favicon.ico" type="image/x-icon">
```

```
<link rel="shortcut icon"  
href="http://www.sitename.com/dirname/favicon.ico"  
type="image/x-icon">
```



HTML: заголовок, общий пример

Теперь рассмотрим общий пример типичного заголовка HTML:

```
<head>
  <title>PHP: Hypertext Preprocessor</title>
  <style type="text/css" media="all">
    @import url("http://static.php.net/www.php.net/styles/site.css");
    @import url("http://static.php.net/www.php.net/styles/phpnet.css");
  </style>

  <!--[if IE]><![if gte IE 6]><![endif]-->
    <style type="text/css" media="print">
      @import url("http://static.php.net/www.php.net/styles/print.css");
    </style>
  <!--[if IE]><![endif]><![endif]-->
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <link rel="shortcut icon" href="http://static.php.net/www.php.net/favicon.ico" />
  <link rel="search" type="application/opensearchdescription+xml"
href="http://www.php.net/phpnetimprovedsearch.src" title="Add PHP.net search" />
  <script type="text/javascript"
src="http://static.php.net/www.php.net/userprefs.js"></script>
  <link rel="alternate" type="application/atom+xml" title="PHP: Hypertext Preprocessor"
href="http://www.php.net/feed.atom" />
</head>
```

HTML: заголовок, общий пример

Итак, по частям...

`<head>` и `</head>` – открывающая и закрывающая части тега **HEAD** соответственно. Между ними находится сам заголовок.

`<title>`PHP: Hypertext Preprocessor`</title>` – название страницы.

`<style type="text/css" media="all"> ... </style>` – указание на описание стилей.

`@import url("http:// ... /styles/site.css");` – способ «спрятать» CSS от старых браузеров, при этом оставив его доступным для новых браузеров.

HTML: заголовок, общий пример

Конструкции вида `<!--[if IE]><![if gte IE 6]><![endif]-->` представляют собой «условия в CSS», т.е. позволяют указывать ту или иную инструкцию CSS в зависимости от вида и версии браузера, например:

`<!--[if IE 6]>`

здесь пишутся инструкции CSS для Internet Explorer 6

`<![endif]-->`

Подробнее о таких конструкциях – в теме про CSS.

`<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>` – указание типа и кодировки документа.

`<link rel="shortcut icon" href="http:// ... /favicon.ico" />` – установка «иконки сайта».

HTML: заголовок, общий пример

`<link rel="search" type="application/opensearchdescription+xml" href="http://.../phpnetimprovedsearch.src" title="Add PHP.net search" />` — ссылка на т.н. «поисковую страницу», т.е. страницу, с помощью которой осуществляется поиск по данной странице.

`<script type="text/javascript" src="http://static.php.net/www.php.net/userprefs.js"></script>` — ссылка на файл, содержащий код на JavaScript.

`<link rel="alternate" type="application/atom+xml" title="PHP: Hypertext Preprocessor" href="http://www.php.net/feed.atom" />` — ссылка на представление информации с сайта в виде «новостной ленты» (обычно — в виде RSS или Atom).

HTML: заголовок, простой пример

На практике не всегда нужны такие сложные заголовки, можно обойтись и минимальным набором элементов:

```
<head>  
  <title>Название страницы</title>  
  <link rel="stylesheet" href="/global.css" type="text/css">  
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>  
  <link rel="shortcut icon" href="/favicon.ico" />  
  <script type="text/javascript" src="/js/userprefs.js"></script>  
  <link rel="alternate" type="application/rss+xml" title="Наши новости в RSS" href="/rss.php" />  
</head>
```

HTML: данные перед тегом HTML

Иногда в начале HTML-документа можно увидеть несто подобное:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0  
Strict//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Согласно спецификациям HTML и XHTML тег DOCTYPE («объявление типа документа») сообщает клиенту, какую именно версию (X)HTML использовать для анализа и отображения страницы.

Этот тег должен всегда находиться в первой строке каждой страницы. Тег DOCTYPE – важный момент для страниц, претендующих на соответствие стандартам: без него код может не пройти проверку валидаторами.

Тег DOCTYPE, в атрибутах которого указывается полный URI (полный web-адрес), сообщает клиентам, что страницу нужно вывести с соблюдением определенного стандарта или подвида этого стандарта.

HTML: теги тела документа, текст

Основными и наиболее часто используемыми тегами для форматирования текста являются:

`
`

`<h1> </h1> ... <h6></h6>`

`<p></p>`

`<pre></pre>`

`<hr />`

При использовании CSS актуальны теги:

`<div></div>`

``

HTML: теги тела документа, текст, `
`

`
` – непарный тег, разрывает строку в том месте, в котором встречается:

какой-то текст 1`
` какой-то текст 2

```
какой-то текст 1  
какой-то текст 2
```

HTML: теги тела документа, текст, <h1>

<h1></h1> .. <h6></h6> – парные теги, определяют заголовки (H1 – самый большой, H6 – самый маленький). В настоящее время почти всегда применяются с «доопределением» через CSS.

Пример:

<h1>Заголовок 1</h1>

<h6>Заголовок 6</h6>

Заголовок 1

Заголовок 6

HTML: теги тела документа, текст, <p>

<p></p> – парный тег, определяет абзац. Из атрибутов этого тега чаще всего применяется атрибут **align**, определяющий выравнивание текста:

<p align="center">Текст текст ... текст</p>

<p align="left"> Текст текст ... текст </p>

<p align="right"> Текст текст ... текст </p>

<p align="justify"> Текст текст ... текст </p>

Текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст
текст текст текст текст текст текст текст текст текст текст текст текст текст

Текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст
текст текст текст текст текст текст текст текст текст текст текст текст текст

Текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст
текст текст текст текст текст текст текст текст текст текст текст текст текст

Текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст
текст текст текст текст текст текст текст текст текст текст текст текст текст

HTML: теги тела документа, текст, <hr />

<hr /> – непарный тег, рисует горизонтальную линию. Часто используется с атрибутами width, color, align:

```
<hr width="100%" color="#990099" align="center"/>
```



HTML: теги тела документа, текст, <pre>

<pre></pre> – парный тег, определяет текст, который должен быть отображён «с исходным форматированием» (т.е. табуляции, пробелы и переносы строк будут учтены):

```
<pre>А вот
в этом тексте      всё учитывается!
    И пробелы      ,      и табуляции</pre>
```

```
А вот
в этом тексте      всё учитывается!
    И пробелы      ,      и табуляции
```


HTML: теги тела документа, текст, <div>

`<div></div>` – парный тег, определяет «блочный элемент» страницы (который может содержать в т.ч. и текст). Как правило, не используется без CSS:

```
<style type="text/css">
.block1 {
  width: 200px;
  background: #ccc;
  padding: 5px;
  padding right: 20px;
  border: solid 1px black;
  float: left;
}
.block2 {
  width: 200px;
  background: #fc0;
  padding: 5px;
  border: solid 1px black;
  float: left;
  position: relative;
  top: 40px;
  left: - 70px;
}
</style>
```

```
<div class="block1">Текст 1
текст 1 текст 1 Текст 1 текст 1
текст 1Текст 1 текст 1 текст
1Текст 1 текст 1 текст 1Текст 1
текст 1 текст 1</div>
```

```
<div class="block2">Текст 2
текст 2 текст 2 Текст 2 текст 2
текст 2 Текст 2 текст 2 текст 2
Текст 2 текст 2 текст 2</div>
```

Текст 1 текст 1 текст 1
Текст 1 текст 1 текст
1Текст 1 текст 1
1Текст 1 текст 1
1Текст 1 текст 1

Текст 2 текст 2 текст 2
Текст 2 текст 2 текст 2
Текст 2 текст 2 текст 2
Текст 2 текст 2 текст 2

HTML: теги тела документа, текст,

`` – парный тег, определяет «строковый элемент» страницы (который, обычно, содержит текст). Как правило, не используется без CSS:

```
<style type="text/css">
  BODY {
    font-family: Arial, sans-serif;
  }
  .letter {
    color: red;
    font-size: 200%; font-family: serif;
    position: relative;
    top: 5px;
  }
</style>
```

```
<p><span class="letter">Т</span>екст 1 текст 1 текст 1 Текст 1
текст 1 текст 1Текст 1 текст 1 текст 1Текст 1 текст 1 текст
1Текст 1 текст 1 текст 1</p>
```

```
<p><span class="letter">И</span> ещё текст, ещё текст, ещё
текст, ещё текст, ещё текст, ещё текст, ещё текст</p>
```

Текст 1 текст 1 текст 1 Текст 1 текст 1 текст 1Текст 1 текст
1 текст 1Текст 1 текст 1 текст 1Текст 1 текст 1 текст 1

И ещё текст, ещё текст, ещё текст, ещё текст, ещё текст,
ещё текст, ещё текст

HTML: теги тела документа, списки

В HTML чаще всего используются два вида списков:
маркированный (**UL**);
нумерованный (**OL**).

Есть ещё и словарный список (**DL**), который практически не используется обычными пользователями для оформления текстов, а потому довольно часто применяется верстальщиками для оформления шаблонов страниц с доопределением поведения этого тега через CSS (так снижается вероятность того, что теги, использованные верстальщиком для оформления страницы, где-то «вступят в конфликт» с тегами, использованными пользователем (администратором) сайта для оформления текстового наполнения страниц).

HTML: теги тела документа, списки,

Маркированный список состоит из тегов **UL** (определяют сам список) и тегов **LI** (определяют элемент списка). Необязательный атрибут `type` определяет вид маркера и может принимать значения **disc** (по умолчанию), **circle**, **square**:

```
<ul>
  <li>Элемент первый</li>
  <li>Элемент второй</li>
  <li>Элемент третий</li>
</ul>
```

```
<ul type="disc">
  <li>Элемент первый</li>
  <li>Элемент второй</li>
  <li>Элемент третий</li>
</ul>
```

```
<ul type="circle">
  <li>Элемент первый</li>
  <li>Элемент второй</li>
  <li>Элемент третий</li>
</ul>
```

```
<ul type="square">
  <li>Элемент первый</li>
  <li>Элемент второй</li>
  <li>Элемент третий</li>
</ul>
```

- Элемент первый
- Элемент второй
- Элемент третий

- Элемент первый
- Элемент второй
- Элемент третий

- Элемент первый
- Элемент второй
- Элемент третий

- Элемент первый
- Элемент второй
- Элемент третий

HTML: теги тела документа, списки,

Нумерованный список состоит из тегов **OL** (определяют сам список) и тегов **LI** (определяют элемент списка). Необязательный атрибут `type` определяет вид нумерации и может принимать значения **1** (по умолчанию), **A**, **a**, **I**, **i**. Необязательный атрибут `start` определяет первый номер:

```
<ol type="1" start="5">  
<li>Яблоки</li>  
<li>Груши</li>  
<li>Сливы</li>  
</ol>
```

```
<ol type="A" start="5">  
<li>Яблоки</li>  
<li>Груши</li>  
<li>Сливы</li>  
</ol>
```

```
<ol type="a" start="5">  
<li>Яблоки</li>  
<li>Груши</li>  
<li>Сливы</li>  
</ol>
```

```
<ol type="I" start="5">  
<li>Яблоки</li>  
<li>Груши</li>  
<li>Сливы</li>  
</ol>
```

```
<ol type="i" start="5">  
<li>Яблоки</li>  
<li>Груши</li>  
<li>Сливы</li>  
</ol>
```

5. Яблоки
6. Груши
7. Сливы

E. Яблоки
F. Груши
G. Сливы

e. Яблоки
f. Груши
g. Сливы

V. Яблоки
VI. Груши
VII. Сливы

v. Яблоки
vi. Груши
vii. Сливы

HTML: теги тела документа, списки, <dl>

Словарный список состоит из тегов **DL** (определяют сам список), тегов **DT** (определяют элемент списка «термин») и **DD** (определяет элемент списка «определение термина»):

<dl>

<dt>Термин 1</dt>

<dd>Определение термина 1</dd>

<dt>Термин 2</dt>

<dd>Определение термина 2</dd>

</dl>

Термин 1

Определение термина 1

Термин 2

Определение термина 2

HTML: теги тела документа, списки, влож.

Списки могут быть вложенными друг в друга в произвольном порядке на произвольную глубину:

```
<ul>
<li>Эл1, ур1</li>
<li>Эл2, ур1
<ul>
<li>Эл1, ур2</li>
<li>Эл2, ур2</li>
</ul>
</li>
<li>Эл3, ур1
<ol>
<li>Эл1b, ур2</li>
<li>Эл2b, ур2</li>
</ol>
</li>
</ul>
```

- Эл1, ур1
- Эл2, ур1
 - Эл1, ур2
 - Эл2, ур2
- Эл3, ур1
 - 1. Эл1b, ур2
 - 2. Эл2b, ур2

HTML: теги тела документа, списки, маркер

Иногда возникает потребность заменить маркер списка на что-то специфическое. Маркер, отличный от трёх «стандартных» можно задать с использованием свойства CSS `list-style-type`, а картинку в качестве маркера можно задать с использованием свойства CSS `list-style-image`:

```
<style type="text/css">
ul { list-style-type: lower-greek }
</style>
```

```
<ul>
  <li>Элемент 1</li>
  <li>Элемент 2</li>
</ul>
```

α. Элемент 1
β. Элемент 2

```
<style type="text/css">
ul {list-style-image: url(/img/arr.gif) }
</style>
```

```
<ul>
  <li>Элемент 1</li>
  <li>Элемент 2</li>
</ul>
```

► Элемент 1
► Элемент 2

HTML: теги тела документа, оформление

Для оформления текста чаще всего используются следующие теги:

``, `<i></i>`, `<u></u>`, ``, `<tt></tt>`

`<small></small>`, `<big></big>`

``, ``

`<blockquote></blockquote>`, `<cite></cite>`, `<code></code>`

`<ins></ins>`, ``

`<xmp></xmp>`

HTML: теги тела документа, оформление

Тег **B** используется для выделения текста «жирным»:

Просто текст ``жирный текст`` просто текст

Просто текст **жирный текст** просто текст

Тег **I** используется для выделения текста «курсивом»:

Просто текст `<i>`наклонный текст`</i>` просто текст

Просто текст *наклонный текст* просто текст

Тег **U** используется для выделения текста «подчёркнутым»:

Просто текст `<u>`подчёркнуто`</u>` просто текст

Просто текст подчёркнуто просто текст

HTML: теги тела документа, оформление

Тег **STRONG** также используется для выделения текста «жирным», однако он является т.н. «тегом логического форматирования» и его воздействие на текст не строго определено стандартом:

Просто текст ``жирный`` просто текст

Просто текст **жирный** просто текст

Тег **TT** используется для написания текста «моноширинным шрифтом»:

Просто текст `<tt>`моноширинный`</tt>` просто текст

Просто текст моноширинный просто текст

HTML: перекрытие тегов

Использование тегов B, I, U очень наглядно иллюстрирует порой возникающую НЕДОПУСТИМУЮ ситуацию, которая называется «перекрытие тегов» (и является грубейшим нарушением стандарта XHTML).

Допустим, нам нужно оформить текст в виде:

Текст 1 **Текст 2** *Текст 3* Текст 4 Текст 5

НЕПРАВИЛЬНАЯ запись тегов будет такой:

Текст 1 ****Текст 2 **<i>**Текст 3**** Текст 4**</i>** Текст 5

Текст 1 ****Текст 2 **<i>**Текст 3**** Текст 4**</i>** Текст 5

ПРАВИЛЬНАЯ запись тегов будет такой:

Текст 1 ****Текст 2******<i>******Текст 3**** Текст 4**</i>** Текст 5

Текст 1 ****Текст 2******<i>******Текст 3**** Текст 4**</i>** Текст 5

HTML: теги тела документа, оформление

Тег **SMALL** используется для выделения текста «чуть меньшим шрифтом, чем окружающий текст»:

Просто текст `<small>`меньше`</small>` просто текст

Просто текст меньше просто текст

Тег **BIG** используется для выделения текста «чуть большим шрифтом, чем окружающий текст»:

Просто текст `<big>`больше`</big>` просто текст

Просто текст больше просто текст

HTML: теги тела документа, оформление

Тег **SUB** используется для написания подстрочных индексов:

`C₂H₅OH`

C_2H_5OH

Тег **SUP** используется для написания надстрочных индексов:

`x² + x³`

$x^2 + x^3$

HTML: теги тела документа, оформление

Тег **BLOCKQUOTE** используется для выделения цитат отдельным блоком:

Текст `<blockquote>`цитата`</blockquote>` просто текст

Текст

цитата

просто текст

Тег **CITE** также используется для выделения цитат шрифтом:

Просто текст `<cite>`цитата`</cite>` просто текст

Просто текст *цитата* просто текст

Тег **CODE** используется для выделения кода на языках программирования (пишется моноширинным шрифтом):

Просто текст `<code>`\$b=17;`</code>` просто текст

Просто текст \$b=17; просто текст

HTML: теги тела документа, оформление

Тег **INS** используется для выделения «нового текста»:

Старый текст `<ins>`новый текст`</ins>` старый текст

Старый текст новый текст старый текст

Тег **DEL** используется для выделения «старого текста» (зачёркивания):

Новый текст ``старый текст`` новый текст

Новый текст ~~старый текст~~ новый текст

HTML: теги тела документа, оформление

Тег **XMP** используется для «отмены обработки инструкций HTML» на некотором участке текста:

Этот код будет показан «as is»: `<xmp>111</xmp>`

Этот код будет показан "as is":

```
<b>111</b>
```

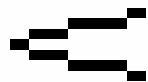
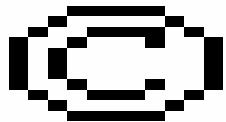
HTML: теги тела документа, оформление

Для оформления текста также используются специальные символы, задаваемые своими кодами или псевдонимами. Их полный список следует смотреть в спецификации HTML.

Синтаксис записи этих символов таков:
&код_или_псевдоним;

В качестве примера рассмотрим:

© < > " —



HTML: теги тела документа, ссылки

Для указания ссылок в документе используется тег `A`. Рассмотрим пример:

```
<a href="http://www.site.com" title="Сайтик" target="_blank">Это наш новый сайт</a>
```

Здесь:

- `href="http://www.site.com"` – URL, по которому нужно перейти;
- `title="Сайтик"` – всплывающая подсказка, которая появится при наведении мыши на ссылку;
- `target="_blank"` – указание, что ссылку нужно открыть в новом окне;
- Это наш новый сайт – текст ссылки, который будет показан пользователю в браузере.

HTML: теги тела документа, ссылки

Атрибут `target` может принимать следующие значения:

`_blank` – открыть в новом окне;

`_top` – открыть без фреймов (если они были);

`_parent` – открыть в родительском фрейме (если есть фреймы);

`_self` – открыть в этом же окне или в этом же фрейме (если есть фреймы) (**используется по умолчанию**);

ИМЯФРЕЙМА – открыть в указанном фрейме (если есть фреймы и есть указанный фрейм).

HTML: теги тела документа, ссылки

Ссылку на почтовый адрес можно оформить так (с использованием ключевого слова **mailto**):

```
<a href="mailto:pupkin@mail.com">Vasya Pupkin</a>
```

При клике по такой ссылке будет запущен «почтовый клиент по умолчанию» и создано пустое письмо, адресатом которого будет pupkin@mail.com

Однако, использование такого подхода категорически не рекомендуется. **Как вы думаете, почему?**

HTML: теги тела документа, ссылки

Ссылку на часть документа можно оформить с использованием части URL, которая называется «сегмент» и пишется в самом конце URL после символа #

Например:

```
<a href="page2.html#chapter4">Click me!</a>
```

При клике по такой ссылке браузер откроет страницу и «прокрутит окно» до той части документа, в которой встретится конструкция:

```
<a name="chapter4"></a>
```

HTML: теги тела документа, картинки

Для включения в HTML-документ картинок используется тег **IMG**, у которого есть следующие наиболее часто используемые атрибуты:

src – путь к графическому файлу;

align – выравнивание и обтекание текстом;

alt – альтернативный текст для изображения;

title – всплывающая подсказка;

border – толщина рамки вокруг изображения;

hspace – горизонтальный отступ;

vspace – вертикальный отступ;

height – высота изображения;

width – ширина изображения.

HTML: теги тела документа, оформление

Пример использования тега **IMG**:

```

```

linux linux linux linux li

linux linux linux linux li

linux linu



linux linu

linux linu

linux linu

linux linu

linux linux linux linux li

• • • • •

HTML: теги тела документа, таблицы

Для оформления таблиц используются следующие теги: TABLE (определяет саму таблицу), TR (определяет ряд таблицы), TD (определяет ячейку таблицы). Пример:

```
<table border="1">
```

```
<tr>
```

```
<td>1</td>
```

```
<td>2</td>
```

```
</tr>
```

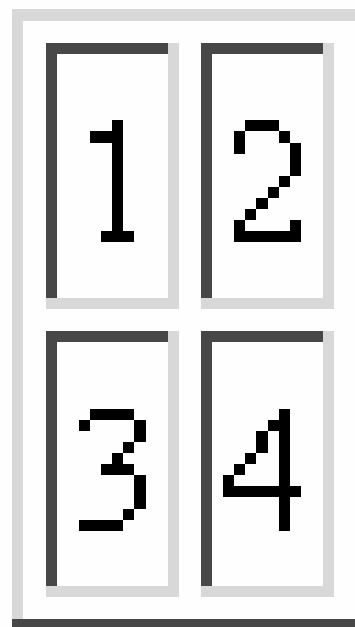
```
<tr>
```

```
<td>3</td>
```

```
<td>4</td>
```

```
</tr>
```

```
</table>
```



1	2
3	4

HTML: теги тела документа, таблицы

Наиболее часто используемые атрибуты тега **TABLE**:

align – горизонтальное выравнивание таблицы;

valign – вертикальное выравнивание таблицы;

width – ширина таблицы (в пикселах или процентах);

height – высота таблицы (в пикселах или процентах);

border – ширина границы таблицы (в пикселах);

cellspacing – расстояние между ячейками (в пикселах);

cellpadding – отступ от границы ячейки до её содержимого (в пикселах);

bgcolor – цвет фона;

background – картинка-фон;

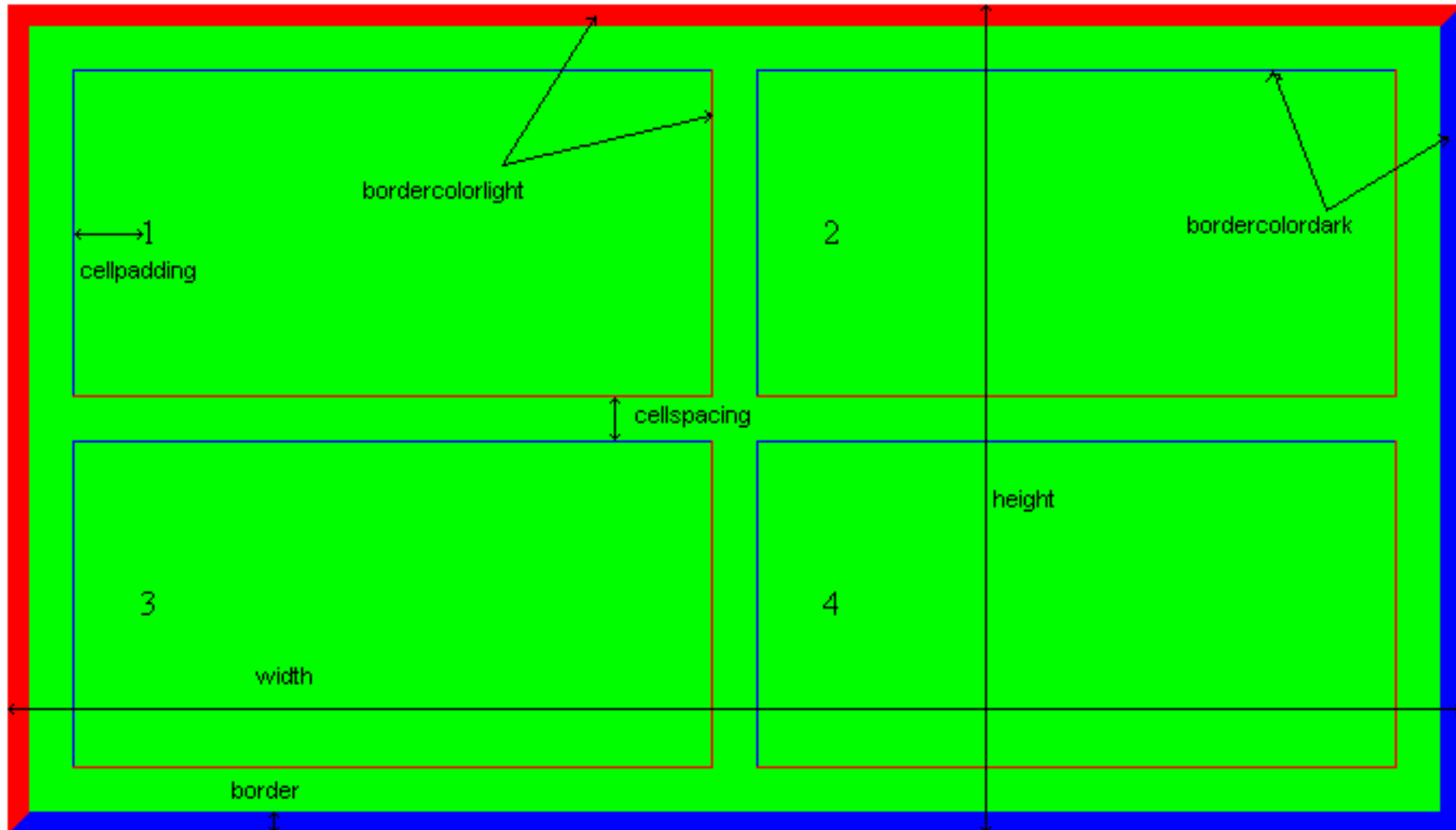
bordercolorlight – цвет «светлой границы» (*);

bordercolordark – цвет «тёмной границы» (*).

(*) поддерживаются не всеми браузерами.

HTML: теги тела документа, таблицы

Рассмотрим пример таблицы:



HTML: теги тела документа, таблицы

Код для получения такой таблицы:

```
<table border="10" width="100%" height="100%"
align="center" valign="top" cellspacing="20" cellpadding="30"
bgcolor="#00FF00" bordercolorlight="#FF0000"
bordercolordark="#0000FF">
  <tr>
    <td>1</td>
    <td>2</td>
  </tr>
  <tr>
    <td>3</td>
    <td>4</td>
  </tr>
</table>
```

HTML: теги тела документа, таблицы

Для задания ряда таблицы применяется тег **TR**. Его наиболее часто используемые атрибуты: **height**, **valign**, **bgcolor**. Модифицируем пример, добавив управление верхним рядом таблицы:

```
<table border="10" width="100%" height="100%" align="center"
valign="top" cellspacing="20" cellpadding="30" bgcolor="#00FF00"
bordercolorlight="#FF0000" bordercolordark="#0000FF">
```

```
<tr height="200" valign="top" bgcolor="#005500">
```

```
<td>1</td>
```

```
<td>2</td>
```

```
</tr>
```

```
<tr>
```

```
<td>3</td>
```

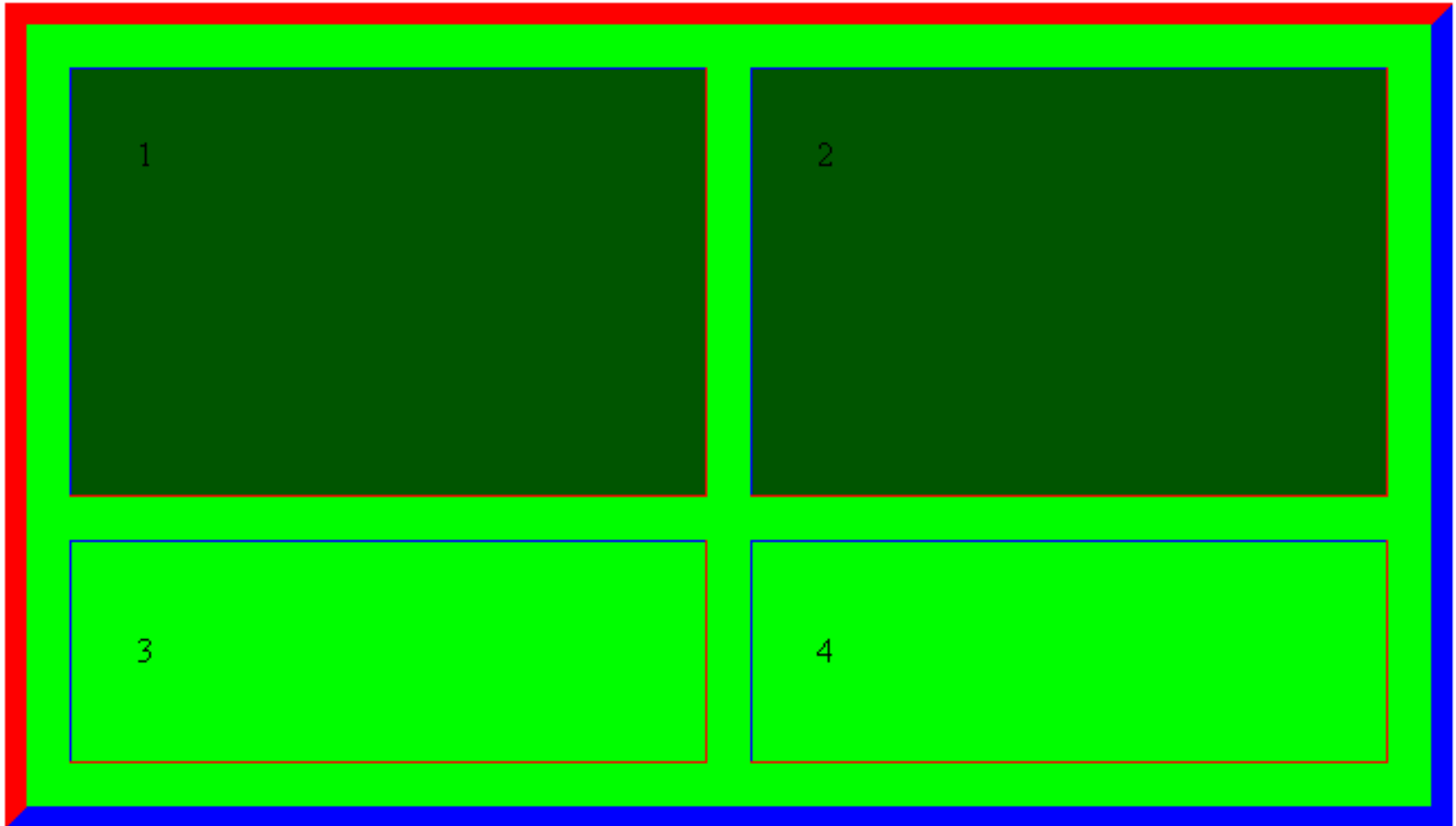
```
<td>4</td>
```

```
</tr>
```

```
</table>
```

HTML: теги тела документа, таблицы

Результат:



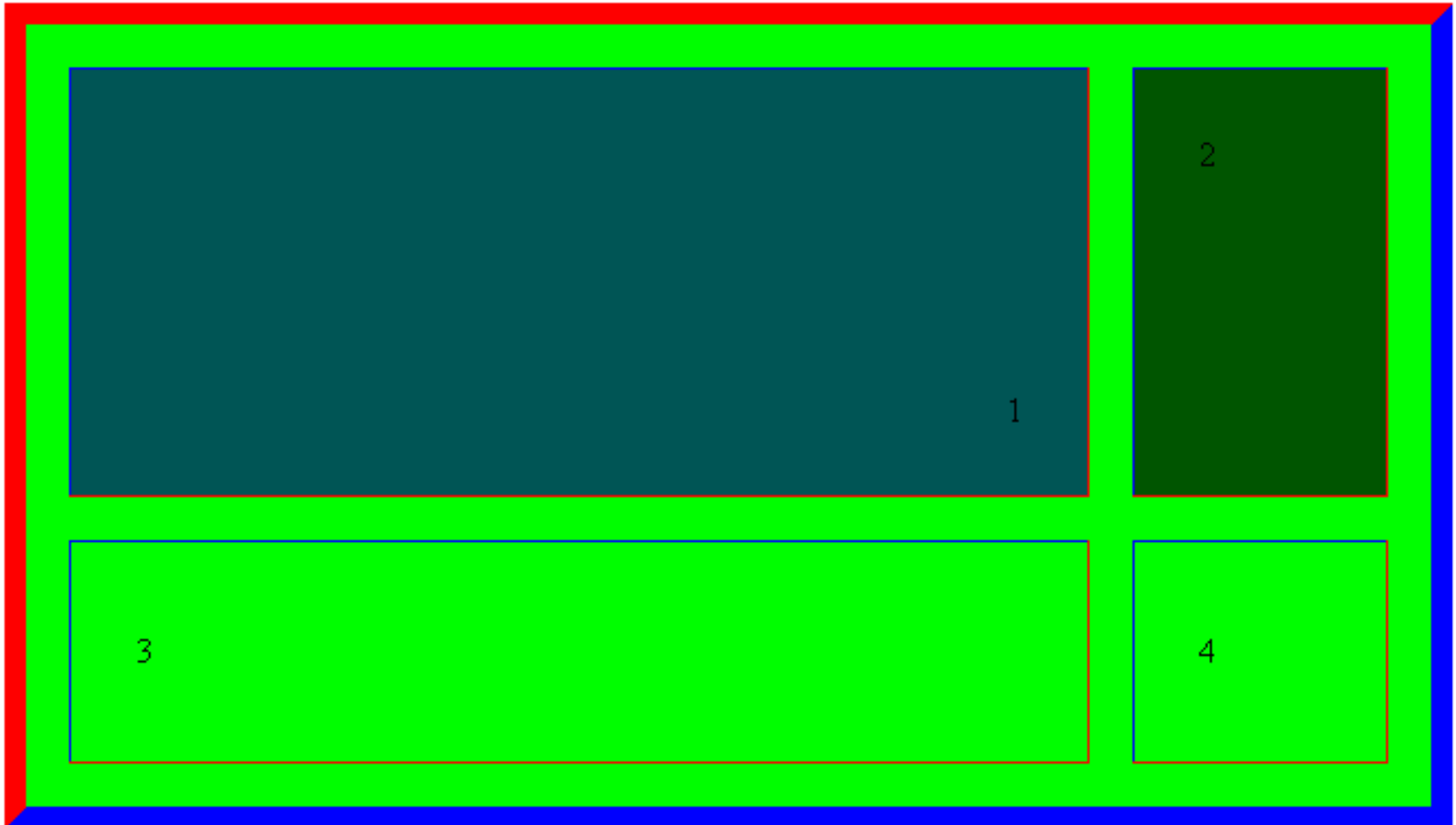
HTML: теги тела документа, таблицы

Для задания ячейки таблицы применяется тег **TD**. Его наиболее часто используемые атрибуты: **width**, **align**, **valign**, **bgcolor**. Модифицируем пример, добавив управление верхней левой ячейкой таблицы:

```
<table border="10" width="100%" height="100%" align="center"
valign="top" cellspacing="20" cellpadding="30" bgcolor="#00FF00"
bordercolorlight="#FF0000" bordercolordark="#0000FF">
  <tr height="200" valign="top" bgcolor="#005500">
    <td valign="bottom" align="right" width="80%"
bgcolor="#005555">1</td>
    <td>2</td>
  </tr>
  <tr>
    <td>3</td>
    <td>4</td>
  </tr>
</table>
```

HTML: теги тела документа, таблицы

Результат:



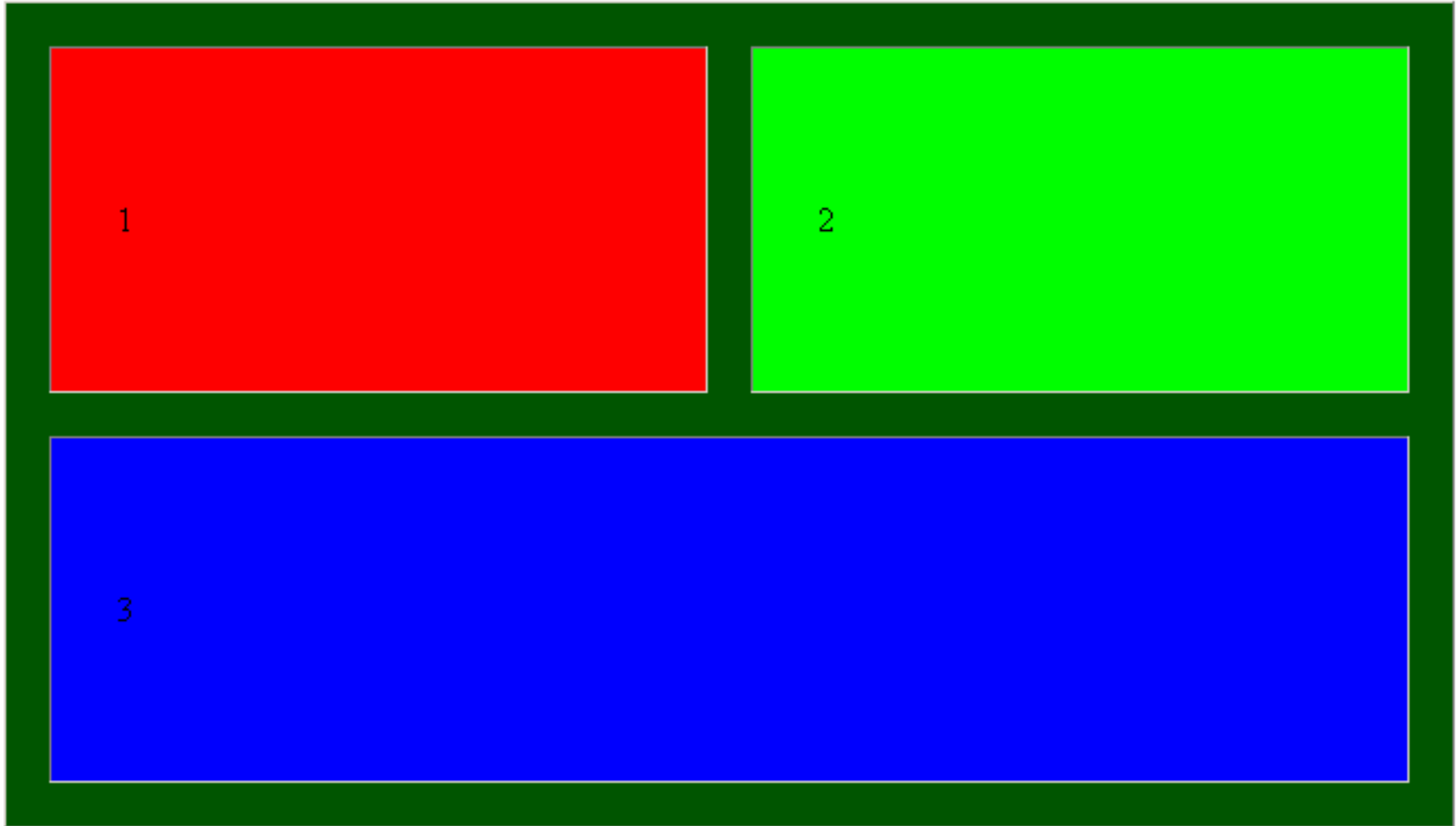
HTML: теги тела документа, таблицы

Для объединения колонок таблицы используется атрибут `colspan`:

```
<table border="1" width="100%" height="100%"
align="center" valign="top" cellpadding="20"
cellpadding="30" bgcolor="#005500">
  <tr>
    <td bgcolor="#FF0000">1</td>
    <td bgcolor="#00FF00">2</td>
  </tr>
  <tr>
    <td colspan="2" bgcolor="#0000FF">3</td>
  </tr>
</table>
```

HTML: теги тела документа, таблицы

Результат:



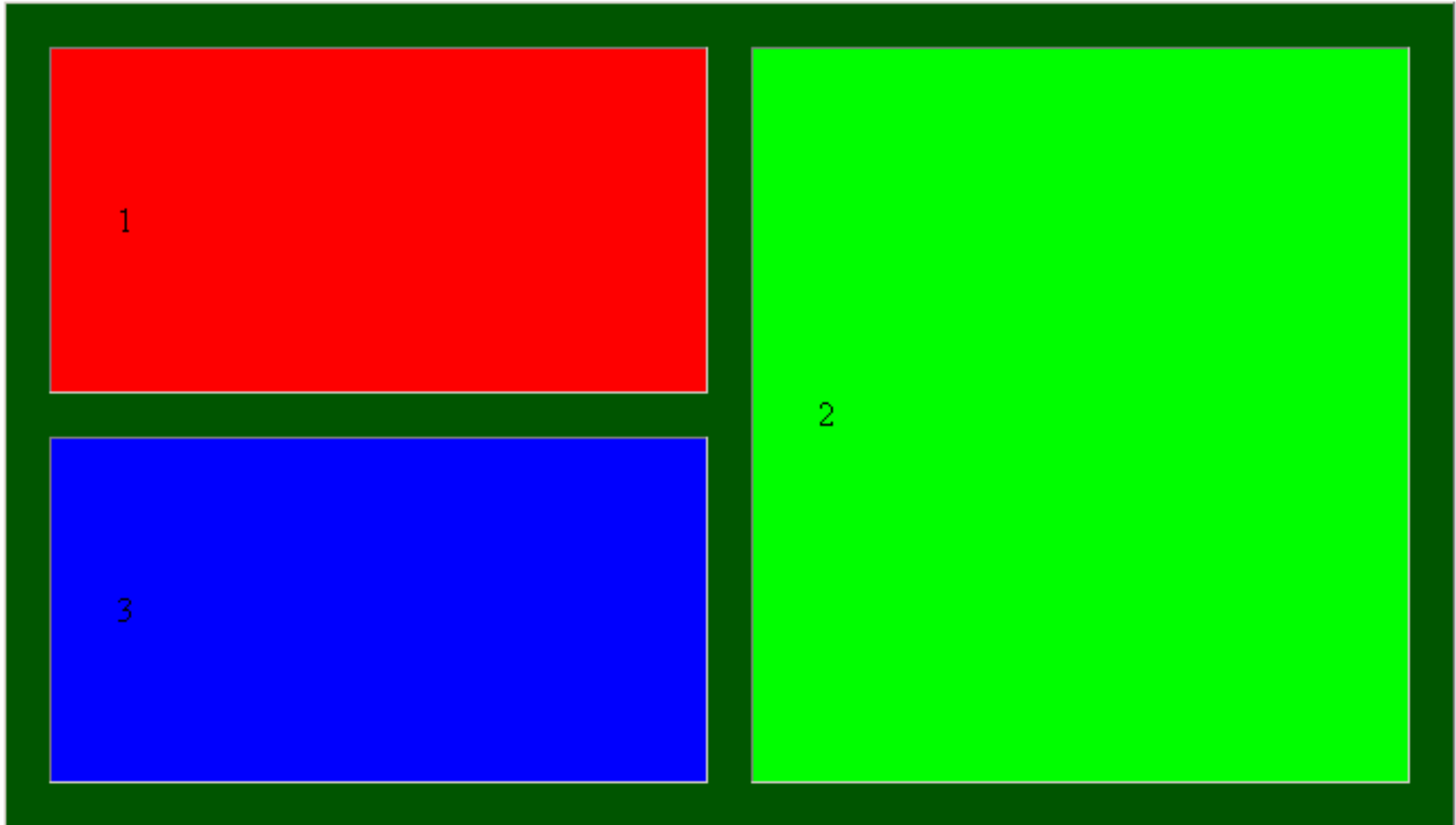
HTML: теги тела документа, таблицы

Для объединения строк таблицы используется атрибут `rowspan`:

```
<table border="1" width="100%" height="100%"
align="center" valign="top" cellspacing="20"
cellpadding="30" bgcolor="#005500">
  <tr>
    <td bgcolor="#FF0000">1</td>
    <td bgcolor="#00FF00" rowspan="2">2</td>
  </tr>
  <tr>
    <td bgcolor="#0000FF">3</td>
  </tr>
</table>
```

HTML: теги тела документа, таблицы

Результат:



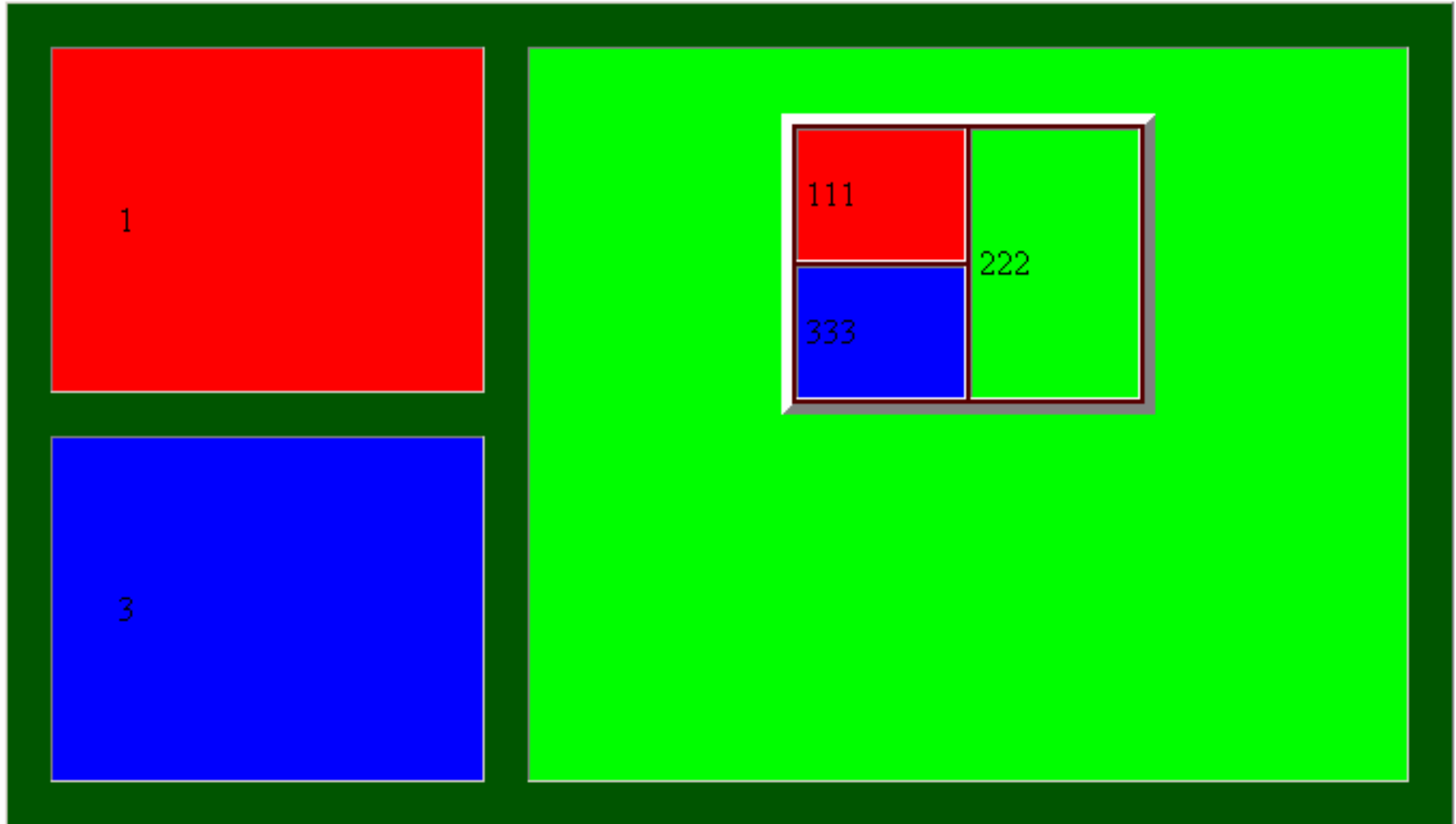
HTML: теги тела документа, таблицы

Таблицы могут быть вложенными на произвольную глубину, однако, делать более 3-5 уровней вложенности не рекомендуется. Пример:

```
<table border="1" width="100%" height="100%" align="center" valign="top" cellspacing="20"
cellpadding="30" bgcolor="#005500">
  <tr>
    <td bgcolor="#FF0000">1</td>
    <td bgcolor="#00FF00" rowspan="2">
      <table border="5" width="50%" height="50%" align="center" valign="top" cellspacing="2"
      cellpadding="3" bgcolor="#550000">
        <tr>
          <td bgcolor="#FF0000">111</td>
          <td bgcolor="#00FF00" rowspan="2">222</td>
        </tr>
        <tr>
          <td bgcolor="#0000FF">333</td>
        </tr>
      </table>
    </td>
  </tr>
  <tr>
    <td bgcolor="#0000FF">3</td>
  </tr>
</table>
```

HTML: теги тела документа, таблицы

Результат:



HTML: фон в виде картинки

Несмотря на то, что фон в виде картинки намного удобнее делать с помощью CSS, его можно задать и с помощью атрибута background, который определён для тегов TABLE, TR, TD.

Пример:

```
<table background="/pics/img1.jpg">  
  <tr background="/pics/img2.jpg">  
    <td background="/pics/img3.jpg">1</td>  
  </tr>  
</table>
```

HTML: комментарии

HTML поддерживает комментарии, т.е. последовательности, которые никак не интерпретируются клиентским ПО. Однако помните, что человеку, просматривающему исходный код страницы, они видны.

Синтаксис таков:

<!-- сколь угодно

много

строк

комментария -->

HTML: теги тела документа, формы

Формы используются для получения информации от пользователя, которая может включать: тексты, выбранные «семафоры», файлы и т.п.

Для обозначения форм используется тег **FORM**, наиболее часто используемые атрибуты которого таковы:

action – указывает URL, на который будут переданы данные формы;

method – указывает метод передачи данных (GET или POST);

enctype – указывает способ кодирования передаваемой информации;

name – указывает имя формы (для работы с формой через JavaScript);

target – указывает способ открытия страницы с формой (аналогично тегу **A**).

Применение JavaScript позволяет значительно расширить возможности форм. Об этом мы поговорим немного позднее...

HTML: теги тела документа, формы

Атрибут `method` может принимать значения `GET` или `POST` (ранее существовал метод `MAILTO`, но он не используется уже многие годы).

Метод `GET` приводит к передаче данных формы «через URL», т.е. после отправки формы URL, по которому были переданы данные, примет примерно такой вид:

`somepage.php?name=Vasya&surname=Pupkin&city=Minsk`

Синтаксис передачи методом `GET` таков:

`?имя_переменной=значение&имя_переменной=значение&` и т.д.

Достоинства метода `GET`:

- удобен для отладки.

Недостатки метода `GET`:

- нельзя передать файлы;
- имеются довольно жёсткие ограничения на объём переданных данных (на практике, лучше, чтобы вся длина URL не превышала 255 символов, хотя для большинства браузеров максимум == 2048);
- могут возникнуть проблемы с кодировками;
- «искушает» пользователя «подкорректировать» значения переменных и «посмотреть, что получится».

HTML: теги тела документа, формы

Метод POST приводит к передаче данных формы «через заголовок HTTP-запроса». При этом URL будет «выглядеть по-человечески»:

`somepage.php` (и всё!)

Достоинства метода POST:

- можно передавать файлы;
- значительно менее жёсткие ограничения по объёму переданных данных (теоретически, размер не ограничен);
- значительно меньше проблем с кодировками;
- пользователь (и находящиеся рядом) не видит непосредственно переданных данных, так что пароль, например, подсмотреть не получится);

Недостатки метода POST:

- неудобен для отладки.

Сейчас мы сделаем небольшое отступление от форм и поговорим о протоколе HTTP. Итак...

HTML: о протоколе HTTP

HTTP (HyperText Transfer Protocol, протокол передачи гипертекста) – протокол прикладного уровня (см. информацию о семиуровневой модели ISO/OSI) передачи данных.

HTTP используется также в качестве транспорта для других протоколов прикладного уровня, таких как SOAP. Основным объектом манипуляции в HTTP является ресурс, на который указывает URI (Uniform Resource Identifier) в запросе клиента.

Обычно такими ресурсами являются хранящиеся на сервере файлы, но ими могут быть логические объекты или что-то абстрактное. Особенностью протокола HTTP является возможность указать в запросе и ответе способ представления одного и того же ресурса по различным параметрам: формату, кодировке, языку и т. д.

Именно благодаря возможности указания способа кодирования сообщения клиент и сервер могут обмениваться двоичными данными, хотя данный протокол является текстовым.

HTML: о протоколе HTTP

HTTP – протокол прикладного уровня, аналогичными ему являются FTP, SMTP, POP3 и т.п.

Обмен сообщениями идёт по обыкновенной схеме «запрос-ответ». В отличие от многих других протоколов, HTTP не сохраняет своего состояния. Это означает отсутствие сохранения промежуточного состояния между парами «запрос-ответ».

Компоненты, использующие HTTP, могут самостоятельно осуществлять сохранение информации о состоянии, связанной с последними запросами и ответами (например, для этого используется механизм сессий, о котором будет сказано позже).

HTML: о протоколе HTTP

Достоинства HTTP

Простота. Протокол настолько прост в реализации, что позволяет с лёгкостью создавать не только клиентские приложения, но и примитивные серверы буквально за минуты.

Расширяемость. Можно легко расширять возможности протокола благодаря внедрению своих собственных заголовков, сохраняя совместимость с другими клиентами и серверами. Они будут игнорировать неизвестные им заголовки.

Распространённость. При выборе протокола HTTP для решения конкретных задач немаловажным фактором является его распространённость. Как следствие, это обилие различной документации по протоколу на многих языках мира, включение удобных в использовании средств разработки в популярные IDE, поддержка протокола в качестве клиента многими программами и обширный выбор среди хостинговых компаний с серверами HTTP.

HTML: о протоколе HTTP

Недостатки HTTP

Большой размер сообщений. Использование текстового формата в протоколе порождает соответствующий недостаток: большой размер сообщений по сравнению с передачей двоичных данных. Из-за этого возрастает нагрузка на оборудование при формировании, обработке и передаче сообщений. Для решения данной проблемы в протокол встроены средства для обеспечения кэширования на стороне клиента, а также средства компрессии передаваемого контента.

Отсутствие «навигации». Хотя протокол разрабатывался как средство работы с ресурсами сервера, у него отсутствуют в явном виде средства навигации среди этих ресурсов. Например, клиент не может явным образом запросить список доступных файлов, как в протоколе FTP. Предполагалось, что конечный пользователь уже знает URI необходимого ему документа, получив который, он будет производить навигацию благодаря гиперссылкам. Это вполне нормально и удобно для человека, но затруднительно, когда стоят задачи автоматической обработки и анализа всех ресурсов сервера без участия человека.

Отсутствие поддержки распределённости. Протокол HTTP разрабатывался для решения типичных бытовых задач где само по себе время обработки запроса должно занимать незначительное время или вообще не приниматься в расчёт. Но в промышленном использовании с применением распределённых вычислений при высоких нагрузках на сервер протокол HTTP оказывается беспомощен. В 1998 году W3C предложил альтернативный протокол HTTP-NG (HTTP Next Generation) для полной замены устаревшего с акцентированием внимания именно на этой области. Данный протокол до сих пор находится на стадии разработки.

HTML: о протоколе HTTP

Структура протокола HTTP

Каждое HTTP-сообщение состоит из трёх частей:

- стартовая строка (starting line) – определяет тип сообщения;
- заголовки (headers) – характеризуют тело сообщения, параметры передачи и прочие сведения;
- тело сообщения (message body) – непосредственно данные сообщения (отделяется от заголовка пустой строкой).

Заголовки и тело сообщения могут отсутствовать, но стартовая строка является обязательным элементом, так как указывает на тип запроса/ответа.

HTML: о протоколе HTTP

Структура протокола HTTP (продолжение)

Стартовая строка

Стартовые строки различаются для запроса и ответа. Строка запроса выглядит так:

`GET URI` – для версии протокола 0.9.

`method URI HTTP/version` – для остальных версий.

Метод (method) – метод запроса, одно слово заглавными буквами. В версии HTTP 0.9 использовался только метод GET, список запросов для версии 1.1 представлен ниже.

- URI определяет путь к запрашиваемому документу.
- Версия (version) – пара разделённых точкой арабских цифр. Например: 1.0.

Чтобы запросить страницу клиент должен передать строку: `GET /somedir/somefile.php HTTP/1.0`

Стартовая строка ответа сервера имеет следующий формат:

`HTTP/version status_code reason_phrase`

- Версия (version) – пара разделённых точкой арабских цифр как в запросе.
- Код состояния (status_code) – три арабские цифры. По коду статуса определяется дальнейшее содержимое сообщения и поведение клиента.
- Пояснение (reason_phrase) – текстовое короткое пояснение к коду ответа для пользователя. Никак не влияет на сообщение и является необязательным.

Например, на предыдущий наш запрос клиентом сервер ответил может ответить строкой: `HTTP/1.0 200 Ok`

HTML: о протоколе HTTP

Методы

Метод HTTP (HTTP method) – последовательность из любых символов (кроме управляющих) и разделителей, указывающая на основную операцию над ресурсом. Обычно метод представляет собой короткое английское слово записанное заглавными буквами.

Обратите внимание что название метода чувствительно к регистру. Каждый сервер обязан поддерживать как минимум методы **GET** и **HEAD**. Если сервер не распознал указанный клиентом метод, то он должен вернуть статус **501 (Not Implemented)**.

Если серверу метод известен, но он не применим к конкретному ресурсу, то возвращается сообщение с кодом **405 (Method Not Allowed)**.

В обоих случаях серверу следует включить в сообщение ответа заголовок **Allow** со списком поддерживаемых методов. Кроме методов **GET** и **HEAD** часто применяется метод **POST**.

Рассмотрим список основных методов...

HTML: о протоколе HTTP

Метод OPTIONS. Используется для определения возможностей веб-сервера или параметров соединения для конкретного ресурса. В ответ серверу следует включить заголовок `Allow` со списком поддерживаемых методов. Также в заголовки ответа может включаться информация о поддерживаемых расширениях.

Для того чтобы узнать возможности всего сервера клиент должен указать в URI звёздочку `*`.

Запросы `OPTIONS * HTTP/1.1` могут также применяться для проверки работоспособности сервера (аналогично «пингованию») и тестирования на предмет поддержки сервером протокола HTTP версии 1.1. Результат выполнения этого метода не кэшируется.

HTML: о протоколе HTTP

Метод GET. Используется для запроса содержимого указанного ресурса. С помощью метода GET можно также начать какой-либо процесс. В этом случае в тело ответного сообщения следует включить информацию о ходе выполнения процесса. Клиент может передавать параметры выполнения запроса в URI целевого ресурса после символа ?:

`GET /path/resource?param1=value1¶m2=value2 HTTP/1.1`

Согласно стандарту HTTP, запросы типа GET считаются «идемпотентными» (многократное повторение одного и того же запроса GET должно приводить к одинаковым результатам, при условии, что сам ресурс не изменился за время между запросами). Это позволяет кэшировать ответы на запросы GET.

Кроме обычного метода GET различают ещё условный GET и частичный GET. Условные запросы GET содержат заголовки If-Modified-Since, If-Match, If-Range и подобные.

Частичные GET содержат в запросе Range. Порядок выполнения подобных запросов определён стандартами отдельно.

HTML: о протоколе HTTP

Метод HEAD. Аналогичен методу GET, за исключением того, что в ответе сервера отсутствует тело. Запрос HEAD обычно применяется для извлечения метаданных, проверки наличия ресурса («валидация URL») и чтобы узнать, не изменился ли он с момента последнего обращения.

Заголовки ответа могут кэшироваться. При несовпадении метаданных ресурса с соответствующей информацией в кэше копия ресурса помечается как устаревшая.

Метод POST. Применяется для передачи пользовательских данных заданному ресурсу. Например, в блогах посетители обычно могут вводить свои комментарии к записям в HTML-форму, после чего они передаются серверу методом POST и он помещает их на страницу.

При этом передаваемые данные (в примере с блогами – текст комментария) включаются в тело запроса. Аналогично, с помощью метода POST обычно загружаются файлы.

В отличие от метода GET, метод POST не считается «идемпотентным», то есть многократное повторение одних и тех же запросов POST может возвращать разные результаты.

Сообщение ответа сервера на выполнение метода POST не кэшируется.

HTML: о протоколе HTTP

Метод PUT. Применяется для загрузки содержимого запроса на указанный в запросе URI. Если по заданному URI не существовало ресурса, то сервер создаёт его и возвращает статус 201 (Created). Если же был изменён ресурс, то сервер возвращает 200 (Ok) или 204 (No Content). Сервер не должен игнорировать некорректные заголовки Content-* передаваемые клиентом вместе с сообщением. Если какой-то из этих заголовков не может быть распознан или не допустим при текущих условиях, то необходимо вернуть код ошибки 501 (Not Implemented).

Фундаментальное различие методов POST и PUT заключается в понимании предназначений URI ресурсов. Метод POST предполагает, что по указанному URI будет производиться обработка передаваемого клиентом содержимого. Используя PUT, клиент предполагает, что загружаемое содержимое будет размещено на ресурсе.

Сообщения ответов сервера на метод PUT не кэшируются.

Метод PATCH. Аналогичен PUT, но применяется только к фрагменту ресурса.

Метод DELETE. Удаляет указанный ресурс.

HTML: о протоколе HTTP

Примеры

HTTP-запрос:

GET /dir/page.php HTTP/1.1

Host: somesite.com

User-Agent: Mozilla/5.0 (X11; U; Linux i686; ru; rv:1.9b5) Gecko/2008050509 Firefox/3.0b5

Accept: text/html

Connection: close

HTTP-ответ:

HTTP/1.0 200 OK

Server: nginx/0.6.31

Content-Language: ru

Content-Type: text/html; charset=utf-8

Content-Length: 1234

Connection: close

На этом мы заканчиваем обсуждение протокола HTTP и возвращаемся к формам...

HTML: теги тела документа, формы

Внутри тега **FORM** используются следующие теги:

input – определяет строковые текстовые поля ввода данных, а также «чекбоксы», «ридиобаттоны» и некоторые кнопки;

select – определяет списки;

option – определяет элементы списка внутри тега **SELECT**;

optgroup – определяет «контейнер» (группу) элементов списка;

textarea – определяет многострочные поля ввода текста;

fieldset – позволяет сгруппировать элементы формы;

legend – определяет заголовок группы элементов формы, заданной с помощью тега **FIELDSET**;

label – определяет связь между меткой (надписью) и элементом формы, определённым тегом **INPUT**;

button – определяет кнопки (их поведение определяется с использованием JavaScript).

HTML: теги тела документа, формы

INPUT — непарный тег, определяющий основные элементы ввода информации.

Главный атрибут тега **INPUT** — **type**:

text — текстовое поле;

password — текстовое поле с паролем;

radio — переключатель типа «радиобаттон»;

checkbox — переключатель типа «чекбокс»;

hidden — скрытое поле;

button — кнопка;

submit — кнопка для отправки формы;

reset — кнопка для восстановления исходных значений полей формы

file — поле для отправки файла;

image — «кнопка с изображением» (её поведение аналогично кнопке **submit**).

Рассмотрим подробнее...

HTML: теги тела документа, формы

```
<input type="text" name="city" readonly="" disabled=""  
size="30" maxlength="20" value="Minsk" />
```

Здесь (**type** и **name** – обязательные атрибуты):

name – имя элемента;

readonly – указание на то, что содержимое поля не может изменяться;

disabled – блокировка изменения состояния элемента;

size – размер элемента в количестве символов;

maxlength – максимальное количество символов, которое можно ввести;

value – значение, которое будет в поле изначально (по умолчанию).

HTML: теги тела документа, формы

```
<input type="password" name="pwd" size="30"
maxlength="20" />
```

Здесь (**type** и **name** – обязательные атрибуты):

name – имя элемента;

size – размер элемента в количестве символов;

maxlength – максимальное количество символов, которое
МОЖНО ВВЕСТИ;



HTML: теги тела документа, формы

```
<input type="checkbox" name="cb1" value="abc" checked="" />
```

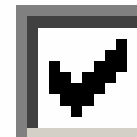
Здесь (**type** и **name** – обязательные атрибуты):

name – имя элемента;

value – значение, которое примет переменная \$cb1, если «чекбокс» будет «выбран».

checked – указание на то, что «чекбокс» должен быть «выбран» изначально (по умолчанию);

Если атрибут **value** не указан и «чекбокс» «выбран», на сервер придёт переменная \$cb1 со значением «on». Если «чекбокс» не «выбран» переменная \$cb1 может вообще не прийти на сервер.



HTML: теги тела документа, формы

```
<input type="radio" name="rb1" value="a" checked="" />  
<input type="radio" name="rb1" value="b" />  
<input type="radio" name="rb1" value="c" />
```

Здесь (**type**, **name** и **value** – обязательные атрибуты):

name – имя элемента;

value – значение, которое примет переменная \$rb1 в зависимости от выбранной «опции».

checked – указание на то, что эта «опция» должна быть «выбрана» изначально (по умолчанию);

Обратите внимание: значение атрибута **name** для «группы радиобаттонов» должно совпадать!



HTML: теги тела документа, формы

```
<input type="submit" name="go" value="Send data!" />
```

Здесь (**type** и **name** – обязательные атрибуты):

name – имя элемента;

value – надпись на кнопке.

Клик по такой кнопке приводит к отправке данных формы на обработку.



HTML: теги тела документа, формы

```
<input type="reset" name="res" value="Set it all back"/>
```

Здесь (`type`, `name` – обязательные атрибуты):

`name` – имя элемента;

`value` – надпись на кнопке.

Клик по такой кнопке приводит к восстановлению значений всех полей формы в том виде, какой они имели сразу после загрузки страницы до каких бы то ни было действий пользователя с формой.

A rectangular button with a light gray background and a thin black border. The text "Set it all back" is centered on the button in a black, monospaced font.

HTML: теги тела документа, формы

```
<input type="image" src="1.jpg" name="gogo" />
```

Здесь (**type**, **src** и **name** – обязательные атрибуты):

name – имя элемента;

src – путь (URL) к картинке.

Клик по такой кнопке-картинке аналогичен клику по кнопке `type="submit"`, т.е. приводит к отправке данных формы на обработку.



HTML: теги тела документа, формы

```
<input type="button" name="btn" value="Click me"/>
```

Здесь (`type`, `name` – обязательные атрибуты):

`name` – имя элемента;

`value` – надпись на кнопке.

Клик по такой кнопке не приводит ни к чему, пока не будет написан соответствующий код на JavaScript.

A rectangular button with a light gray background and a dark gray border. The text "Click me" is centered on the button in a black, monospaced font.

HTML: теги тела документа, формы

```
<input type="file" name="myfile" size="30" />
```


Здесь (**type** и **name** – обязательные атрибуты):

name – имя элемента;

size – размер элемента в количестве символов.

У элемента с **type="file"** свойство **value** является read-only, т.е. его невозможно указать заранее (это сделано в целях повышения безопасности пользователей).

Обратите внимание: чтобы файлы корректно передавались на сервер и обрабатывались там, необходимо указать атрибуту **enctype** тега **form** значение **multipart/form-data** (по умолчанию этот атрибут принимает значение **application/x-www-form-urlencoded**).



HTML: теги тела документа, формы

```
<input type="hidden" name="hid1" value="5" />
```

Здесь (**type**, **name** и **value** – обязательные атрибуты):

name – имя элемента;

value – значение элемента.

Такой элемент никак не отображается визуально, однако бывает полезен для передачи «технических данных».

Обратите внимание: пользователь может просмотреть и изменить эти данные, отредактировав страницу. Т.е. не стоит передавать здесь информацию, которую пользователь не должен видеть и иметь возможность изменить.

HTML: теги тела документа, формы

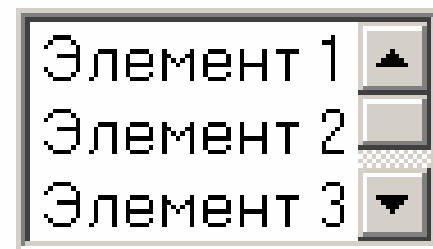
```
<select name="sel" size="1">  
  <option value="AAA">Элемент 1</option>  
  <option value="12">Элемент 2</option>  
</select>
```

Здесь (**name** – обязательный атрибут):

name – имя элемента;

value – «компьютеропольное» значение элемента.

Если значение **size** равно 1 – получится «выпадающий список», если оно более 1 – получится «список с прокруткой» (значение **size** указывает высоту списка).



HTML: теги тела документа, формы

Чтобы указать, какой элемент списка будет выбран по умолчанию, у соответствующего `option` следует прописать атрибут `selected=""`:

```
<option value="12" selected="">Элемент 2</option>
```

У «списков с прокруткой» может быть выбрано несколько элементов, для этого нужно модифицировать описание самого списка:

```
<select size="3" multiple="" name="sel[]">
```

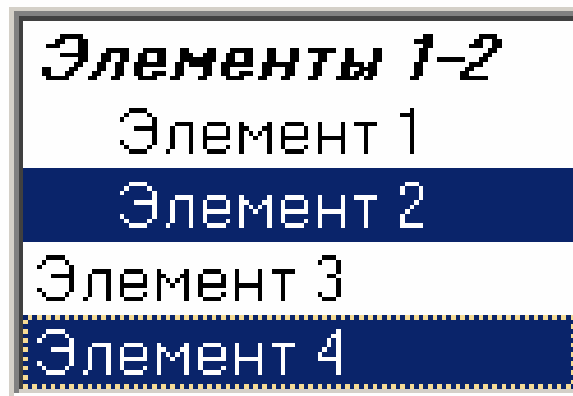
Обратите внимание на то, что имя списка теперь указывается с квадратными скобками. В таком случае на сервер для обработки будет передан массив с именем `$sel`, содержащий те элементы, которые выбраны в списке.



HTML: теги тела документа, формы

Указать группу элементов можно с помощью тега `optgroup`:

```
<select size="5" multiple="" name="sel[]">  
  <optgroup label="Элементы 1-2">  
    <option value="AAA">Элемент 1</option>  
    <option value="12">Элемент 2</option>  
  </optgroup>  
  <option value="123">Элемент 3</option>  
  <option value="124">Элемент 4</option>  
</select>
```



HTML: теги тела документа, формы

Тег `textarea` (парный) определяет многострочные поля ввода текста:

```
<textarea name="ta1" rows="3" cols="10">
```

Какой-то текст

```
</textarea>
```

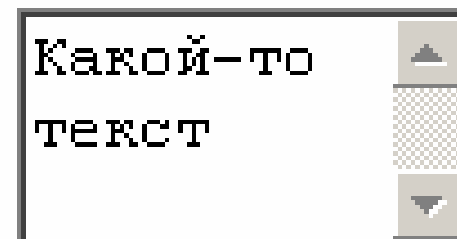
Здесь (`name` – обязательный атрибут):

`name` – имя элемента;

`rows` – высота элемента (в строках);

`cols` – ширина элемента (в колонках).

Обратите внимание, что атрибута `value` у тега `textarea` нет. Если нужно указать значение по умолчанию, оно прописывается между открывающей и закрывающей частями тега.



HTML: теги тела документа, формы

Тег `button` (парный) позволяет создать «просто кнопку» (аналогично `<input type="button" name="btn" value="Click me"/>`). Придать такой кнопке поведение, аналогичное кнопкам отправки данных формы и восстановления данных формы можно с использованием атрибута `type`, который может принимать значения: `button` (по умолчанию), `submit` и `reset`.

`<button>`Кнопка с текстом`</button>`

`<button>`Кнопка с рисунком`</button>`

Клик по такой кнопке не приводит ни к чему, пока не будет написан соответствующий код на JavaScript.

Кнопка с текстом

▶ Кнопка с рисунком

HTML: теги тела документа, формы

Тег `fieldset` (парный) позволяет сгруппировать набор полей для удобства восприятия, а тег `legend` позволяет указать имя группы полей:

`<fieldset>`

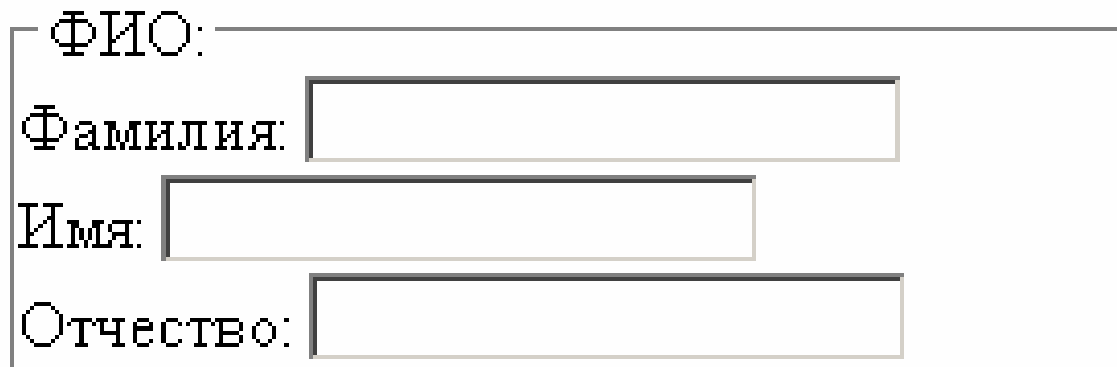
`<legend>ФИО:</legend>`

Фамилия: `<input type="text" name="fio_f" />
`

Имя: `<input type="text" name="fio_n" />
`

Отчество: `<input type="text" name="fio_o" />
`

`</fieldset>`



The image shows a visual representation of the HTML code provided. It features a large rectangular box with a thin border. Inside this box, at the top left, is the text 'ФИО:'. Below this text, there are three horizontal input fields, each preceded by a label: 'Фамилия:', 'Имя:', and 'Отчество:'. The input fields are empty and have a light gray border. The labels and input fields are stacked vertically, with the 'Фамилия:' label and its input field being the topmost, followed by 'Имя:', and then 'Отчество:' at the bottom.

HTML: теги тела документа, формы

Тег `label` (парный) позволяет привязать надпись к элементу формы и установить «горячую клавишу» (срабатывает по комбинации `Alt+указанная_клавиша`) для быстрого перехода к элементу формы:

```
<input type="checkbox" id="check1" accesskey="3" />  
<label for="check1">Получать спам</label>
```

Такой «чекбокс» будет изменять своё состояние при клике как по нём самом, так и по надписи рядом с ним, а также будет становиться «выбранным» при нажатии `Alt+3`.

Обратите внимание, что связь элемента формы и надписи происходит по атрибуту `id` (который, к слову, может присутствовать у почти любого элемента HTML-документа и очень удобен при использовании JavaScript для прямого обращения к элементу).

☐ Получать спам

HTML: теги тела документа, формы

Мы рассмотрели все элементы формы в отдельности. Теперь рассмотрим несколько примеров форм.

Итак, простейшая форма для входа зарегистрированного пользователя:

```
<form action="login.php" method="post">
```

```
  Логин: <input type="text" name="ul" /><br/>
```

```
  Пароль: <input type="password" name="up" /><br/>
```

```
  <input type="submit" name="go" value="Войти" />
```

```
</form>
```

Логин:

Пароль:

HTML: теги тела документа, формы

Теперь та же форма, но свёрстанная «чуть более красиво» и с дополнением в виде «чекбокса»:

```
<form action="login.php" method="post">
<table border="0">
  <tr>
    <td align="right">Логин:</td>
    <td align="left"><input type="text" name="ul" /></td>
  </tr>
  <tr>
    <td align="right">Пароль:</td>
    <td align="left"><input type="password" name="up" /></td>
  </tr>
  <tr>
    <td align="right">Запомнить меня:</td>
    <td align="left"><input type="checkbox" name="remember" /></td>
  </tr>
  <tr>
    <td align="center" colspan="2">
      <input type="submit" name="go" value="Войти" />
    </td>
  </tr>
</table>
</form>
```

HTML: теги тела документа, формы

Результат:

Логин:

Пароль:

Запомнить меня: ☐

Войти

HTML: теги тела документа, формы

Иногда возникает задача передать на сервер для обработки множество однотипных элементов (например, отмеченные для удаления почтовые сообщения). В таком случае эффективно использовать передачу массивов. Для этого нужно указать множество элементов с одним и тем же именем с квадратными скобками, но отличающиеся значением value.

Например:

```
<input type="checkbox" name="cb[]" value="1" />  
<input type="checkbox" name="cb[]" value="2" />  
<input type="checkbox" name="cb[]" value="3" />
```

В случае с «чекбоксами» на сервер придёт массив, состоящий только из значений выбранных «чекбоксов». В случае с другими элементами – придёт массив, содержащий значения всех элементов группы.

HTML: теги тела документа, формы

В таких массивах также можно определять ключи:

```
<input type="checkbox" name="cb[one]" value="1" />
```

```
<input type="checkbox" name="cb[two]" value="2" />
```

```
<input type="checkbox" name="cb[three]" value="3" />
```

При такой записи переменная-массив будет содержать элементы не с ключами `0`, `1`, `2`, а с ключами `one`, `two`, `three`.

Если вы хотите использовать список с возможностью выбора нескольких пунктов, его имя также следует объявлять в виде массива:

```
<select name="sel1[]" multiple="" size="20">
```

HTML: теги тела документа, формы

Формы могут включать сотни элементов, быть «пошаговыми», динамически изменяться с помощью JavaScript и т.д. и т.п. Однако, все они строятся на уже рассмотренных нами принципах.

На этом мы заканчиваем наше рассмотрение HTML.