СПб ГУТ)))

Основы интернет-технологий

WEB-программирование

Основы Web-программирования

ОСНОВНЫЕ ПРИНЦИПЫ СОЗДАНИЯ СТРАНИЦ СРЕДСТВАМИ HTML

Принципы гипертекстовой разметки

- ***HTML*** - описательный *язык разметки* документов
- В нем используются указатели разметки (***теги***)
- ***Теговая*** **модель**: документ как совокупность контейнеров, каждый из которых начинается и заканчивается *тегами.*
- **HTML-** ***тег*** состоит из **имени**, за которым может следовать *список* **атрибутов**. Текст *тега* заключается в угловые скобки ("<" и ">")
- **< … >** - тег, **открывающий** контейнер
- **</ … >** - тег, **закрывающий** контейнер (нет атрибутов)
- Теги могут быть **вложенными**
- Бывают **непарные** теги (img, br, hr)

Атрибуты тега

- **Атрибут** – параметр тега, который может иметь конкретные значения, устанавливаемые автором для изменения функции *тега.*
- **Синтаксис:**

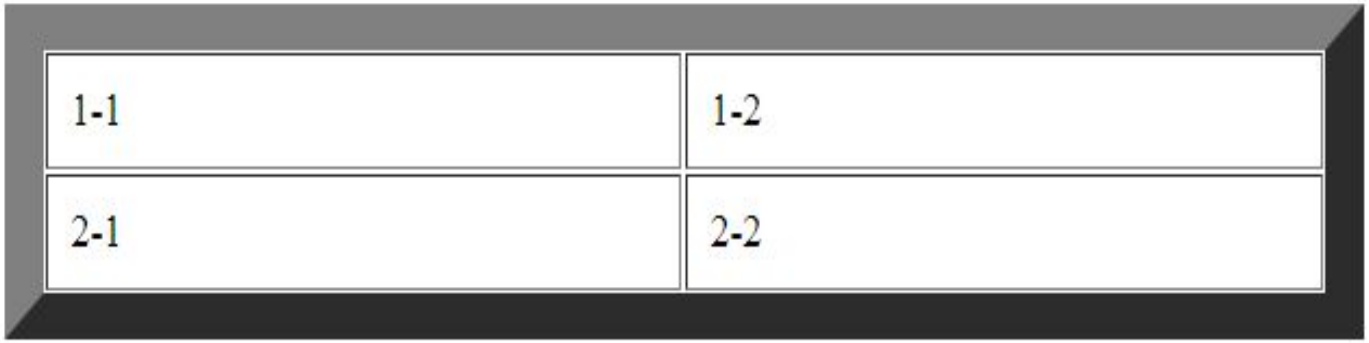
<тег атрибут=“значение”>

- ***Пример**:*

<TABLE WIDTH=570 ALIGN=center CELLPADDING=10 CELLSPACING=2 BORDER=16>

Пример

```
<TABLE  WIDTH=570  ALIGN=center CELLPADDING=10
CELLSPACING=2 BORDER=16>
  <tr>
    <td>1-1</td>
    <td>1-2</td>
  </tr>
  <tr>
    <td>2-1</td>
    <td>2-2</td>
  </tr>
</table>
```



1-1	1-2
2-1	2-2

Группы тегов HTML

Все *теги* HTML *по* их назначению и области действия можно разделить на следующие основные группы:

- определяющие структуру документа;
- оформление блоков гипертекста (параграфы, списки, таблицы, картинки);
- гипертекстовые ссылки и закладки;
- формы для организации диалога;
- вызов программ.

Структура HTML-документа

HTML-документ — это один большой *контейнер*, который начинается с тега <HTML> и заканчивается тегом </HTML>

```
<html>
  <head>
     СОДЕРЖАНИЕ ЗАГОЛОВКА
  </head>
  <body>
     СОДЕРЖАНИЕ ТЕЛА ДОКУМЕНТА
  </body>
</html>
```

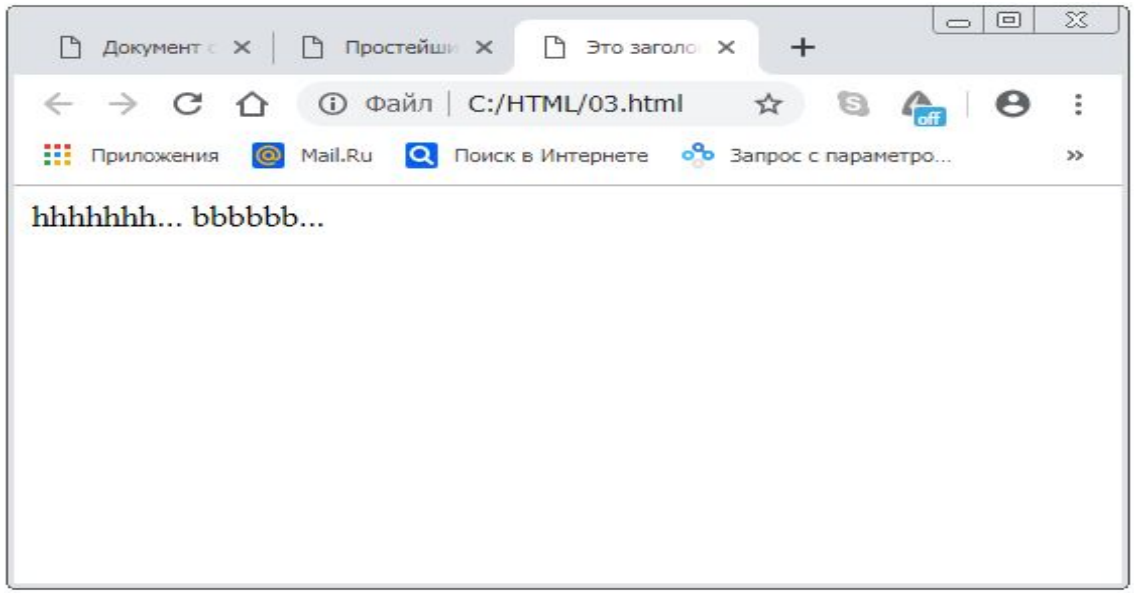
Теги разделов html-документа

- <html> … </html> - парные теги всего документа
- <head> … </head> - теги заголовка документа
- <body> … </body> - теги тела документа

Заголовок документа
<Head>

Необязательный элемент разметки.

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



Основные контейнеры заголовка

- *HEAD* (элемент разметки *HEAD*);
- *TITLE* (заглавие документа);
- *META* (метаинформация);
- *LINK* (общие ссылки);
- *STYLE* (описатели стилей);
- *SCRIPT* (скрипты).

Элемент разметки HEAD

- Элемент разметки HEAD содержит заголовок HTML-документа.
- Атрибутов у тега начала контейнера нет
- Контейнер *заголовка* служит для размещения информации, относящейся ко всему документу в целом.

Элемент разметки TITLE

- Элемент разметки ***TITLE*** служит для именования документа в World Wide Web - именование окна браузера, в котором просматривается документ.
- Необязательный тег
- Парный тег.
- Тег не имеет атрибутов.
- Роботы многих поисковых систем используют содержание элемента *TITLE* для создания поискового образа документа и помещают его в индекс поисковой системы.
- Синтаксис:

<TITLE>название документа</TITLE>

Элемент разметки META

- ***META*** содержит управляющую информацию, которую браузер использует для правильного отображения и обработки содержания тела документа.
- **Непарный** тег.
- **Содержание**:
  1. Кодировки,
  2. Язык документа,
  3. Управление роботами,
  4. Управление кэшом,
  5. Автоматическая загрузка страниц
  6. Служебная информация
  7. Информация об авторе

Атрибуты тега META

- **HTTP-EQUIV.** Конвертирует метатег в заголовок HTTP. Значение этого атрибута преобразовывается в формат заголовка ответа протокола HTTP.
- **NAME.** Имя, по нему определяется предназначение мета тега.
- **CONTENT.** Обязательный атрибут. Устанавливает значение атрибута, заданного с помощью name и http-equiv
- **CHARSET.** Устанавливает кодировку документа.

1. Задание кодировки

- С поддержкой таблиц UNICODE появилась возможность указывать тип кодировки документа — CHARSET.
- Для перекодировки на стороне клиента (документ подготовлен в кодировке ср1251)
  в *заголовок* документа необходимо включить *META*-тег следующего вида:

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

2. Язык документа

Content-Language

Задает язык документа и сообщает о нем роботам, индексирующим страницу. Они могут определять язык и сами, но тег будет полезен в случае сбитых настроек браузера на стороне пользователя.

Пример установки русского языка:

```
<meta
  http-equiv="Content-Language"
  content="ru">
```

Новый вариант атрибута

- Было:

<meta

http-equiv="content-language"

content="ru">

- Стало (приоритет выше):

<html lang="ru">

3. Управление роботами
Описание поискового образа документа

Для описания документа используется два *META* - тега:

- Список ключевых слов (keywords)
- Реферат (description) (краткое содержание документа), который отображается в качестве пояснения к ссылке на документ в отчете *поисковой машины* о выполненном запросе.

Пример реферата и ключ. слов

<META

NAME="description"

HTTP-EQUIV ="description"

CONTENT ="Учебный курс Основы Web-технологий. Тема: Заголовок HTML-документа. Элемент разметки META. Дается краткое описание основных способов применения контейнера META в заголовке HTML-документа. Рассматривается управление HTTP-обменом и индексирование документов.">

<META

NAME="keywords"

HTTP-EQUIV="keywords"

CONTENT="учебный курс; Web-технология; web; технология; HTML; язык гипертекстовой разметки; заголовок HTML-документа; заголовок; HTML; документ; контейнер; META; элемент; HEAD; пример; разметка; методика">

4. Управление кэшем

Cache-Control. Указывает браузеру о действии кэша в отношении данного документа.

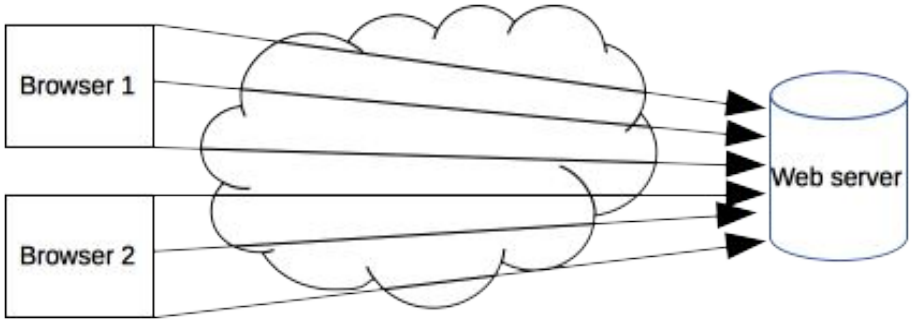
```
<meta
    http-equiv="Cache-Control"
    content ="значение">
```

Значения атрибута content

- public. Кэширование будет возможным во всех доступных кэшах.
- private. Кэшируется только в частном кэше, но не кэшируется прокси-сервером.
- no-cache. Полный запрет кэширования.(HTTP/1.1)
- no-store. Может кэшироваться, но не сохраняется в архиве.
- max-age=time, must-revalidate. Задает максимальное время, в секундах, в течении которого браузер должен хранить страницу в кэше.
- max-age=time, proxy-revalidate. Тоже максимальное время хранения, в секундах, но не для браузера, а для прокси-сервера

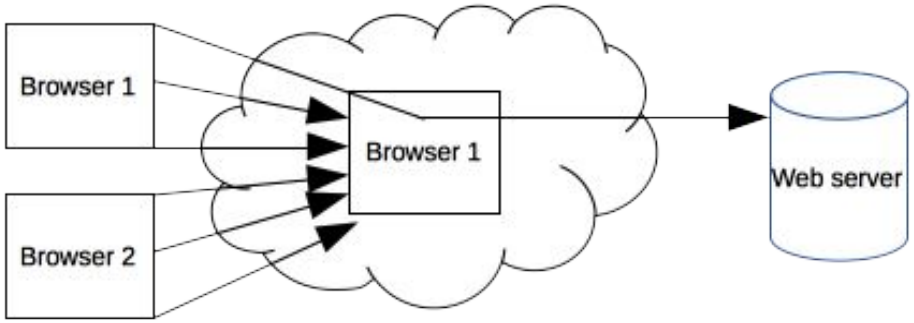
Виды кэширования

No cache



All identical requests are
going through to the server.

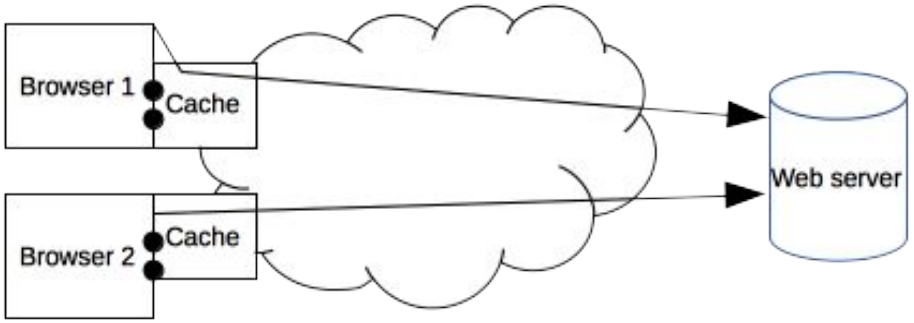
Shared cache



The first request is going through.

Subsequent identical requests are served
by the shared cache.
(more efficient)

Local (private) cache



The first request of each client is going through.

Subsequent identical requests are not even sent, but
served by the local cach.
(most efficient, except for first requests)

Запрет кэширования HTTP/1.0.

Pragma. Запрещает кэширование страниц. Тег может использоваться, если информация размещенная на странице сайта часто обновляется. Он не задан для ответов HTTP. Использовать его следует только для совместимости с клиентами HTTP/1.0 (1996-1999).

Пример:

```
<meta
    http-equiv="pragma"
    content="no-cache"
>
```

Кэширование по расписанию

Expires. Заведует кэшированием документа. Если заявленное в этом значении время прошло, то браузер должен опять запросить у сервера страницу, а не использовать страницу из кэша.

Пример:

```
<meta
    name="expires"
    content="Sat, 16 Mar 2019 08:00:00 GMT "
>
```

5. Принудительная перезагрузка

- REFRESH
- Пример: перезагрузка страницы «01.html» через 5 секунд

```
<META
    HTTP-EQUIV="Refresh"
    CONTENT="5; URL=01.html">
```

Можно построить автоматически перезагружаемую последовательность страниц

Пример

```
<HEAD>
  <META
     HTTP-EQUIV="Refresh"
     CONTENT="1; URL=02.html">
</HEAD>

<BODY  bgcolor=#ff0000>
  СОДЕРЖАНИЕ ТЕЛА ДОКУМЕНТА
</BODY>
```

```
< HEAD >
    <META
          HTTP-EQUIV="Refresh"
          CONTENT="1; URL=01.html">
  </ HEAD >


  < BODY  bgcolor=#0000ff>
    СОДЕРЖАНИЕ ТЕЛА ДОКУМЕНТА
  </ BODY >
```

Служебная информация

- Generator,
- Reply-to,
- Author,
- Copyright.

Generator

- Указывает в какой программе создана веб-страница. Автоматически проставляется, если страница создана с использованием какого-то программного обеспечения.
- Пример:

```
<meta
    name="Generator"
    content="Название генератора">
```

Reply-to

- Рассказывает, как связаться с владельцем веб-сайта, автором текста или еще кем-нибудь. В атрибуте content обычно указывают адрес электронной почты данного лица.

- Пример:

```
<meta
    name="Reply-to"
    content="адрес электронной почты">
```

Author, Copyright

- Идентифицируют личность автора и принадлежность документа. Метатег author содержит ФИО настоящего автора, но если сайт принадлежит компании, то лучше и уместнее всего использовать Copyright.
- Пример:

```
<meta
    name="Author"
    content="Имя автора">
<meta
    name="Copyright"
    content="Название фирмы">
```

Элемент разметки LINK

- Связывание текущего документа с внешними ресурсами (например, со стилями).
- Одинарный тег.
- Синтаксис:

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

- Пример:

```
<link
    rel="stylesheet"
    type="text/css"
    href=«mytheme.css">
```

Элемент разметки STYLE

- Для размещения описателей стилей. Задает правила отображения контейнеров HTML-документа для всей страницы.
- Синтаксис:

```
<STYLE
    TYPE= “text/css”>
    описание стиля/стилей
</STYLE>
```

Элемент разметки SCRIPT

- Служит для размещения кода JavaScript, VBScript.
- Синтаксис:

```
<SCRIPT
    [TYPE=тип_языка_программирования]
    [SRC=URL]>
</SCRIPT>
```

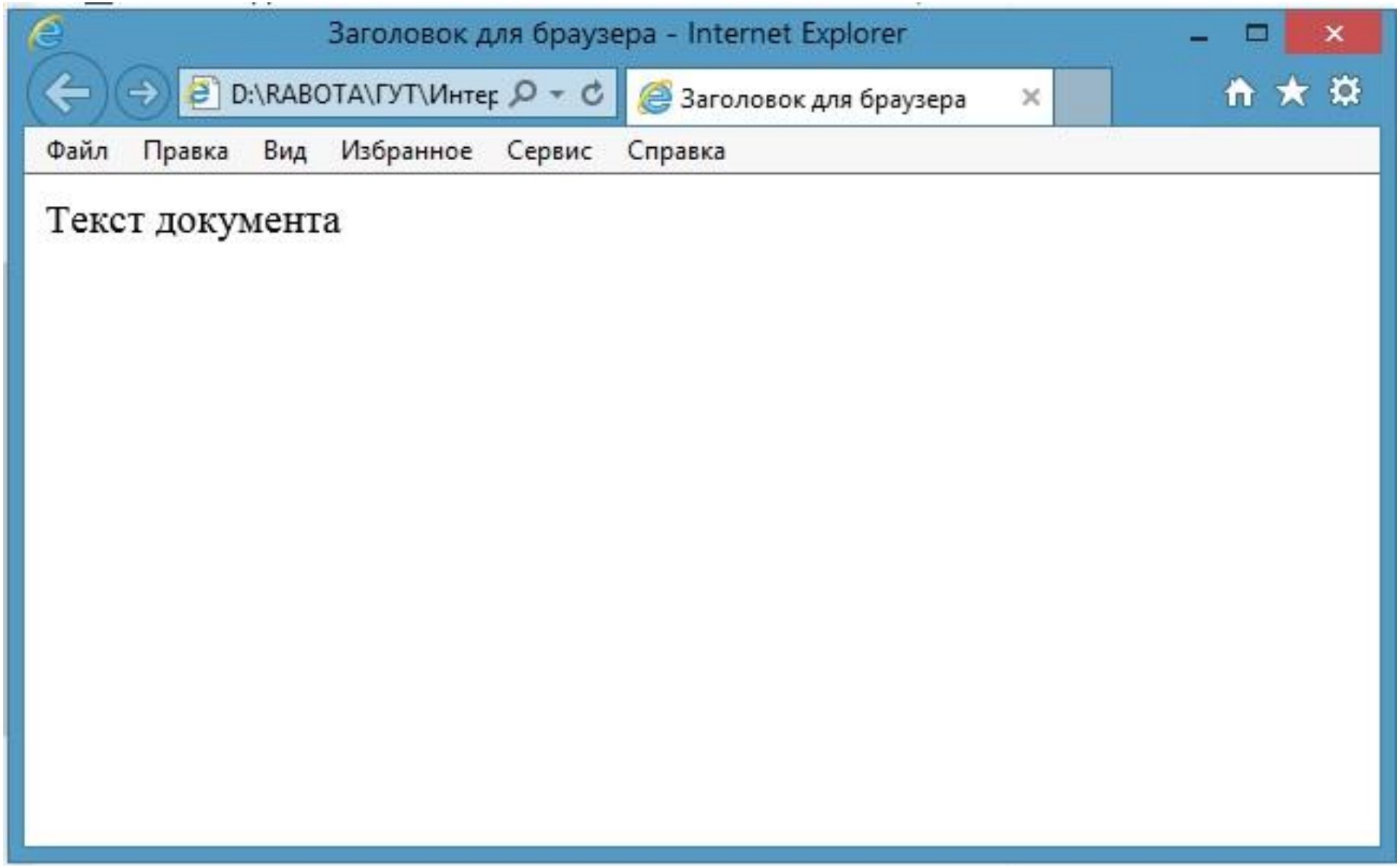
- Пример:

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

Простейшая веб-страница

```
<html>
  <head>
    <title>Заголовок для браузера</title>
    <meta
        http-equiv="Content-Type"
        content="text/html;
        charset=windows-1251">
    <meta
        name="keywords"
        content="веб-страница первая">
  </head>
  <body>
     Текст документа
  </body>
</html>
```

Пример в браузере



Теги тела документа

- Иерархические контейнеры и заставки;
- Заголовки (от H1 до H6);
- *Блоки* (параграфы, списки, формы, таблицы, картинки и т.п.);
- Горизонтальные подчеркивания и адреса;
- Текст, разбитый на области действия стилей (подчеркивание, выделение, курсив);
- Математические описания, графика и гипертекстовые ссылки.

Тело документа – контейнер BODY

- Парный тег
- Имеет атрибуты:
  - Цвета (фона, текста, ссылок (3 варианта))
  - Рисунок фона
  - Отступы от краев окна браузера (4 варианта)
  - Полоса прокрутки

Атрибуты тега body

- **ЦВЕТА**
  - Bgcolor - Цвет фона веб-страницы.
  - Text - Цвет текста в документе.
  - Link - Цвет ссылок на веб-странице.
  - Alink - Устанавливает цвет активной ссылки.
  - Vlink - Цвет посещенных ссылок.

- **РИСУНОК ФОНА СТРАНИЦЫ**
  - Background - Задает фоновый рисунок на веб-странице.

- **ОТСТУПЫ ОТ КРАЕВ**
  - Leftmargin - Отступ от левого края окна браузера до контента.
  - Rightmargin - Отступ от правого края окна браузера до контента.
  - Topmargin - Отступ от верхнего края окна браузера до контента.
  - Bottommargin - Отступ от нижнего края окна браузера до контента.

- **ПРОКРУТКА**
  - Bgproperties - Определяет, прокручивать фон совместно с текстом или нет.
  - Scroll - Устанавливает, отображать полосы прокрутки или нет.

Цветовая схема RGB

- RGB – Red-Green-Blue
- На каждую компоненту цвета отводится

1 байт = 8 бит,

т.е. 8 двоичных цифр или 2 16-ричные цифры

#FF0000 – Красный

#00FF00 – Зеленый

#0000FF – Синий

#000000 – Черный

#FFFFFF – Белый

Некоторые составные цвета

Название	Код	Название	Код
aqua	#00FFFF	navy	#000080
black	#000000	olive	#808000
blue	#0000FF	purple	#800080
fuchsia	#FF00FF	red	#FF0000
gray	#808080	silver	#C0C0C0
green	#008000	teal	#008080
lime	#00FF00	white	#FFFFFF
maroon	#800000	yellow	#FFFF00

Красные тона:

IndianRed	#CD5C5C	205, 92, 92
LightCoral	#F08080	240, 128, 128
Salmon	#FA8072	250, 128, 114
DarkSalmon	#E9967A	233, 150, 122
LightSalmon	#FFA07A	255, 160, 122
Crimson	#DC143C	220, 20, 60
Red	#FF0000	255, 0, 0
FireBrick	#B22222	178, 34, 34
DarkRed	#8B0000	139, 0, 0

Розовые тона:

Pink	#FFC0CB	255, 192, 203
LightPink	#FFB6C1	255, 182, 193
HotPink	#FF69B4	255, 105, 180
DeepPink	#FF1493	255, 20, 147
MediumVioletRed	#C71585	199, 21, 133
PaleVioletRed	#DB7093	219, 112, 147

Оранжевые тона:

LightSalmon	#FFA07A	255, 160, 122
Coral	#FF7F50	255, 127, 80
Tomato	#FF6347	255, 99, 71
OrangeRed	#FF4500	255, 69, 0
DarkOrange	#FF8C00	255, 140, 0
Orange	#FFA500	255, 165, 0

Зелёные тона:

GreenYellow	#ADFF2F	173, 255, 47
Chartreuse	#7FFF00	127, 255, 0
LawnGreen	#7CFC00	124, 252, 0
Lime	#00FF00	0, 255, 0
LimeGreen	#32CD32	50, 205, 50
PaleGreen	#98FB98	152, 251, 152
LightGreen	#90EE90	144, 238, 144
MediumSpringGreen	#00FA9A	0, 250, 154
SpringGreen	#00FF7F	0, 255, 127
MediumSeaGreen	#3CB371	60, 179, 113
SeaGreen	#2E8B57	46, 139, 87
ForestGreen	#228B22	34, 139, 34
Green	#008000	0, 128, 0
DarkGreen	#006400	0, 100, 0
YellowGreen	#9ACD32	154, 205, 50
OliveDrab	#6B8E23	107, 142, 35
Olive	#808000	128, 128, 0
DarkOliveGreen	#556B2F	85, 107, 47
MediumAquamarine	#66CDAA	102, 205, 170
DarkSeaGreen	#8FBC8F	143, 188, 143
LightSeaGreen	#20B2AA	32, 178, 170
DarkCyan	#008B8B	0, 139, 139
Teal	#008080	0, 128, 128

Жёлтые тона:

Gold	#FFD700	255, 215, 0
Yellow	#FFFF00	255, 255, 0
LightYellow	#FFFFE0	255, 255, 224
LemonChiffon	#FFFACD	255, 250, 205
LightGoldenrodYellow	#FAFAD2	250, 250, 210
PapayaWhip	#FFEFD5	255, 239, 213
Moccasin	#FFE4B5	255, 228, 181
PeachPuff	#FFDAB9	255, 218, 185
PaleGoldenrod	#EEE8AA	238, 232, 170
Khaki	#F0E68C	240, 230, 140
DarkKhaki	#BDB76B	189, 183, 107

Синие тона:		
Aqua	#00FFFF	0, 255, 255
Cyan	#00FFFF	0, 255, 255
LightCyan	#E0FFFF	224, 255, 255
PaleTurquoise	#AFEEEE	175, 238, 238
Aquamarine	#7FFFD4	127, 255, 212
Turquoise	#40E0D0	64, 224, 208
MediumTurquoise	#48D1CC	72, 209, 204
DarkTurquoise	#00CED1	0, 206, 209
CadetBlue	#5F9EA0	95, 158, 160
SteelBlue	#4682B4	70, 130, 180
LightSteelBlue	#B0C4DE	176, 196, 222
PowderBlue	#B0E0E6	176, 224, 230
LightBlue	#ADD8E6	173, 216, 230
SkyBlue	#87CEEB	135, 206, 235
LightSkyBlue	#87CEFA	135, 206, 250
DeepSkyBlue	#00BFFF	0, 191, 255
DodgerBlue	#1E90FF	30, 144, 255
CornflowerBlue	#6495ED	100, 149, 237
MediumSlateBlue	#7B68EE	123, 104, 238
RoyalBlue	#4169E1	65, 105, 225
Blue	#0000FF	0, 0, 255
MediumBlue	#0000CD	0, 0, 205
DarkBlue	#00008B	0, 0, 139
Navy	#000080	0, 0, 128
MidnightBlue	#191970	25, 25, 112

Фиолетовые тона:

Lavender	#E6E6FA	230, 230, 250
Thistle	#D8BFD8	216, 191, 216
Plum	#DDA0DD	221, 160, 221
Violet	#EE82EE	238, 130, 238
Orchid	#DA70D6	218, 112, 214
Fuchsia	#FF00FF	255, 0, 255
Magenta	#FF00FF	255, 0, 255
MediumOrchid	#BA55D3	186, 85, 211
MediumPurple	#9370DB	147, 112, 219
BlueViolet	#8A2BE2	138, 43, 226
DarkViolet	#9400D3	148, 0, 211
DarkOrchid	#9932CC	153, 50, 204
DarkMagenta	#8B008B	139, 0, 139
Purple	#800080	128, 0, 128
Indigo	#4B0082	75, 0, 130
SlateBlue	#6A5ACD	106, 90, 205
DarkSlateBlue	#483D8B	72, 61, 139

Коричневые тона:		
Cornsilk	#FFF8DC	255, 248, 220
BlanchedAlmond	#FFEBCD	255, 235, 205
Bisque	#FFE4C4	255, 228, 196
NavajoWhite	#FFDEAD	255, 222, 173
Wheat	#F5DEB3	245, 222, 179
BurlyWood	#DEB887	222, 184, 135
Tan	#D2B48C	210, 180, 140
RosyBrown	#BC8F8F	188, 143, 143
SandyBrown	#F4A460	244, 164, 96
Goldenrod	#DAA520	218, 165, 32
DarkGoldenRod	#B8860B	184, 134, 11
Peru	#CD853F	205, 133, 63
Chocolate	#D2691E	210, 105, 30
SaddleBrown	#8B4513	139, 69, 19
Sienna	#A0522D	160, 82, 45
Brown	#A52A2A	165, 42, 42
Maroon	#800000	128, 0, 0

Белые тона:		
White	#FFFFFF	255, 255, 255
Snow	#FFFAFA	255, 250, 250
Honeydew	#F0FFF0	240, 255, 240
MintCream	#F5FFFA	245, 255, 250
Azure	#F0FFFF	240, 255, 255
AliceBlue	#F0F8FF	240, 248, 255
GhostWhite	#F8F8FF	248, 248, 255
WhiteSmoke	#F5F5F5	245, 245, 245
Seashell	#FFF5EE	255, 245, 238
Beige	#F5F5DC	245, 245, 220
OldLace	#FDF5E6	253, 245, 230
FloralWhite	#FFFAF0	255, 250, 240
Ivory	#FFFFF0	255, 255, 240
AntiqueWhite	#FAEBD7	250, 235, 215
Linen	#FAF0E6	250, 240, 230
LavenderBlush	#FFF0F5	255, 240, 245
MistyRose	#FFE4E1	255, 228, 225

Серые тона:		
Gainsboro	#DCDCDC	220, 220, 220
LightGrey	#D3D3D3	211, 211, 211
LightGray	#D3D3D3	211, 211, 211
Silver	#C0C0C0	192, 192, 192
DarkGray	#A9A9A9	169, 169, 169
DarkGrey	#A9A9A9	169, 169, 169
Gray	#808080	128, 128, 128
Grey	#808080	128, 128, 128
DimGray	#696969	105, 105, 105
DimGrey	#696969	105, 105, 105
LightSlateGray	#778899	119, 136, 153
LightSlateGrey	#778899	119, 136, 153
SlateGray	#708090	112, 128, 144
SlateGrey	#708090	112, 128, 144
DarkSlateGray	#2F4F4F	47, 79, 79
DarkSlateGrey	#2F4F4F	47, 79, 79
Black	#000000	0, 0, 0

Пример задания цветов

```
<BODY
        BGCOLOR=#FFFFFF
        TEXT=#0000FF
        VLINK=#FF0000
        LINK=#00FF00
        BACKGROUND="image.gif"
>
```

Теги управления разметкой

- Заголовки: h1-h6
- Абзац (параграф): p
  - Выравнивание: align
  - Обтекание графики текстом
- Перевод строки: br
- Запрет перевода строки: nobr

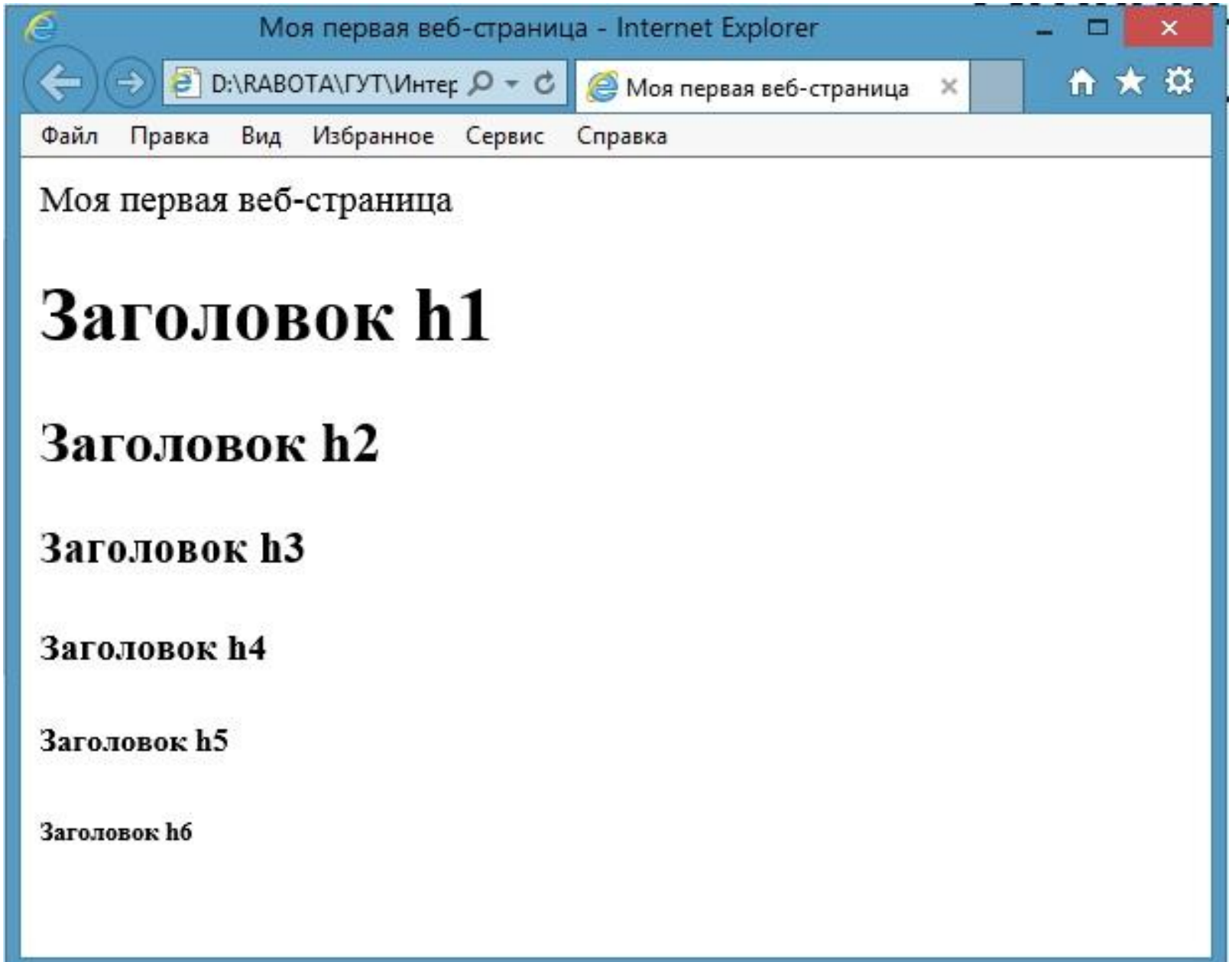
Теги заголовков и абзацев

- …   - теги абзацев
- <h1> … </h1> - теги заголовков,
- <h2> … </h2>,
- <h3> … </h3>, вплоть до <h6> … </h6>
- Число у тега h – степень вложенности

Пример заголовков и абзацев

```
<body>
    <p>Моя первая веб-страница</p>
    <h1>Заголовок h1</h1>
    <h2>Заголовок h2</h2>
    <h3>Заголовок h3</h3>
    <h4>Заголовок h4</h4>
    <h5>Заголовок h5</h5>
    <h6>Заголовок h6</h6>
</body>
```

Пример в браузере



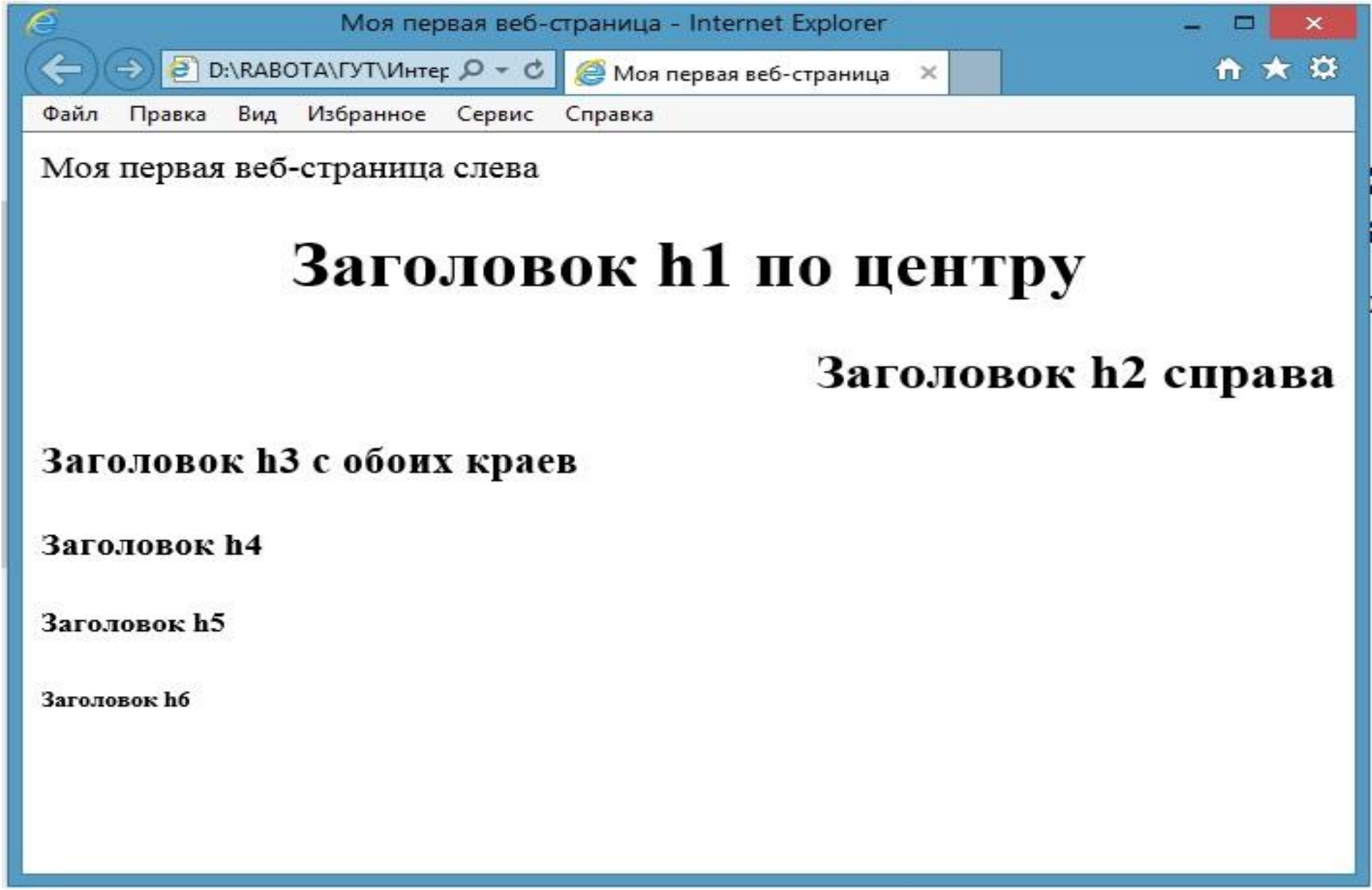
Атрибут выравнивания align

- по левому краю: **ALIGN=left**
- правому краю: **ALIGN=right**
- по центру: **ALIGN=center**
- по ширине: **ALIGN=justify**

Атрибуты форматирования

```
<p align="left">
    Моя первая веб-страница слева
</p>
<h1 align="center">
    Заголовок h1 по центру
</h1>
<h2 align="right">
    Заголовок h2 справа
</h2>
<h3 align="justify">
    Заголовок h3 с обоих краев
</h3>
```

Форматирование в браузере



Обтекание графики текстом

```
<body>
<img src="03.jpg" align="left" hspace="50" vspace="8">
Чебура́шка — персонаж, придуманный писателем Эдуардом Успенским в 1966 году как один из главных героев книги «Крокодил Гена и его друзья» и её продолжений. После выхода мультфильма Романа Качанова «Крокодил Гена», снятого по этой книге в 1969 году,…
</body>
```

Атрибуты HSPACE и VSPACE задает ширину горизонтальных и *вертикальных полей*, отделяющих изображение от текста. Можно также создать рамку вокруг картинки или обрамление таблицы текстом.

Пример обтекания картинки текстом

Чебура?шка — персонаж, придуманный писателем Эдуардом Успенским в 1966 году как один из главных героев книги «Крокодил Гена и его друзья» и её продолжений. После выхода мультфильма Романа Качанова «Крокодил Гена», снятого по этой книге в 1969 году, персонаж стал широко известен. В первоначальном варианте книги внешне представлял собой неуклюжее уродливое существо с маленькими ушами и коричневой шерстью, ходящее на задних лапах. Известный сегодня добродушный образ Чебурашки с большими ушами и большими глазами впервые появился в мультфильме «Крокодил Гена» и был создан при непосредственном участии художника-постановщика фильма Леонида Шварцмана. Согласно отредактированному (относительно размера ушей) предисловию к книге «Крокодил Гена и его друзья», Чебурашкой называлась бывшая в детстве у автора книги бракованная игрушка, изображавшая странного зверя: не то медвежонок, не то заяц с большими ушами. Глаза у него были большие и жёлтые, как у филина, голова круглая, заячья, а хвост коротенький и пушистый, такой, какой бывает обычно у маленьких медвежат. По книге, родители автора утверждали, что это — неизвестный науке зверь, который живёт в жарких тропических лесах. Поэтому в основном тексте, героями которого выступают, как утверждает писатель, детские игрушки самого Эдуарда Успенского, Чебурашка действительно является неизвестным тропическим зверьком, который забрался в ящик с апельсинами, уснул там и в результате вместе с ящиком попал в большой город. Директор магазина, в котором открыли ящик, назвал его «Чебурашкой», так как обьевшийся апельсинами зверёк постоянно падал (чебурахался): Он сидел, сидел, смотрел по сторонам, а потом взял да и чебурахнулся со стола на стул. Но и на стуле он долго не усидел — чебурахнулся снова. На пол. — Фу ты, Чебурашка какой! — сказал про него директор магазина, — Совсем не может сидеть на месте! Так наш зверёк и узнал, что его имя — Чебурашка… Версию про бракованную игрушку, изложенную во введении к своей книге, Э. Н. Успенский отвергает, как сочинённую специально для детей. В интервью нижегородской газете Успенский говорит: Я пришел в гости к другу, а его маленькая дочка примеряла пушистую шубу, которая тащилась по полу, <…> Девчонка постоянно падала, запинаясь о шубу. И её отец после очередного падения воскликнул: «Ой, опять чебурахнулась!». Это слово врезалось мне в память, я спросил его значение. Оказалось, что «чебурахнуться» — это значит «упасть». Так и появилось имя моего героя. В «Толковом словаре живого великорусского языка» В. И. Даля описано как слово «чебурахнуться» в значении «упасть», «грохнуться», «растянуться», так и слово «чебурашка», определяемое им в различных диалектах как «шашка бурлацкой лямки, привешенная на хвосте», либо как «ванька-встанька, куколка, которая, как ни кинь её, сама встаёт на ноги». Согласно этимологическому словарю Фасмера «чебура?хнуть» образовано от слов чубуро?к, чапуро?к, чебура?х — «деревянный шар на конце бурлацкой бечевы», тюркского происхождения. Другое родственное слово это «чебырка» — плётка, на конце которой шарик на волосе. Происхождение слова «чебурашка», в смысле игрушки-неваляшки, описанное у Даля, связанно с тем, что многие рыбаки изготавливали такие игрушки из деревянных шаров, которые являлись поплавками для рыболовных сетей, и также назывались чебурашкой.

Специальные теги

- **HR – Горизонтальная линия (непарный)**
- **SUB – Нижний и индекс**
- **SUP – Верхний индекс**
- **BR – Перевод строки (непарный)**
- **BIG – Увеличения шрифта на одну ступень**
- **SMALL – Уменьшение шрифта**
- **&#код – Спец.символы (не тег).**
- **I – Курсив**
- **B – Жирный**
- **U – подчеркивание**
- **S – перечеркивание**
- **Q – текст в двойных кавычках**
- **BLOCKQUOTE – блок цитат** (Тег добавляет поля слева и справа от текста)

Пример спец. тегов

```
<body>
    &#064
    &#094
    <hr size=3>
    <p>S<sub>m</sub>=3x<sup>2</sup>+15</p>
    <br>
    <p>Чебура́шка — персонаж, <big>придуманный писателем
<big>Эдуардом Успенским в 1966 году как один <small>из главных
героев книги <small> «Крокодил Гена и его друзья» и её
продолжений. После выхода мультфильма Романа Качанова
«Крокодил Гена», снятого по этой книге в 1969 году, персонаж
стал широко известен
    </p>
   <hr size=5>
  </body>
```

Пример спец. тегов

@^

$S_m=3x^2+15$

Чебура?шка — персонаж, придуманный писателем Эдуардом Успенским в 1966 году как один из главных героев книги «Крокодил Гена и его друзья» и её продолжений. После выхода мультфильма Романа Качанова «Крокодил Гена», снятого по этой книге в 1969 году, персонаж стал широко известен.

Комментарии

Комментарии *HTML* начинаются с символа

"<!--"

и оканчиваются символом

"-->"

СПИСКИ

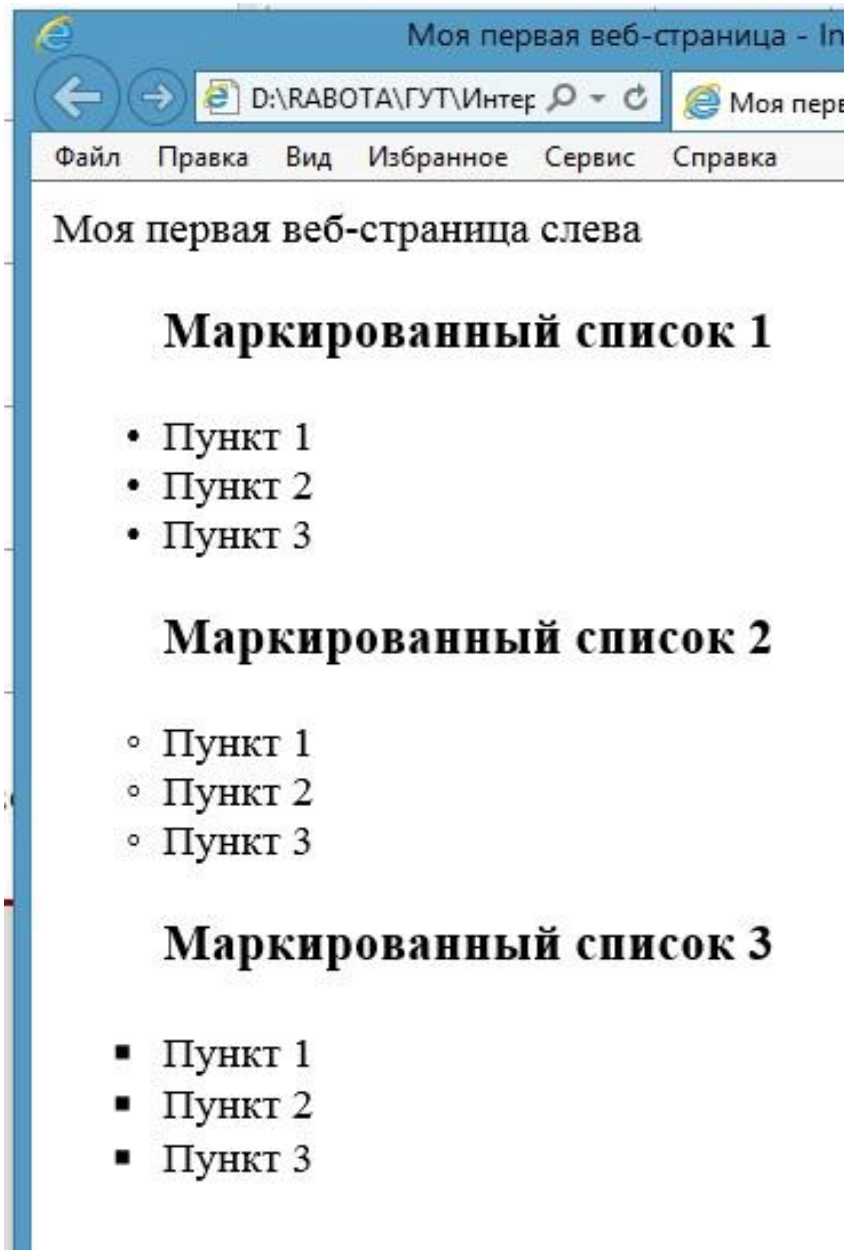
- Маркированные
- Нумерованные
- Определений

Маркированные списки

- … - задание списков
- … - задание элементов списка
- type – атрибут типа маркера
  - disc - закрашенный круг (по умолчанию)
  - circle – незакрашенный круг
  - square – закрашенный квадрат

Пример маркированного списка

```
<p align="left">Моя первая веб-страница слева</p>
<ul type="disc">
  <h3>Маркированный список 1</h3>
  <li>Пункт 1</li>
  <li>Пункт 2</li>
  <li>Пункт 3</li>
</ul>
<ul type="circle">
  <h3>Маркированный список 2</h3>
  <li>Пункт 1</li>
  <li>Пункт 2</li>
  <li>Пункт 3</li>
</ul>
<ul type="square">
  <h3>Маркированный список 3</h3>
  <li>Пункт 1</li>
  <li>Пункт 2</li>
  <li>Пункт 3</li>
</ul>
```

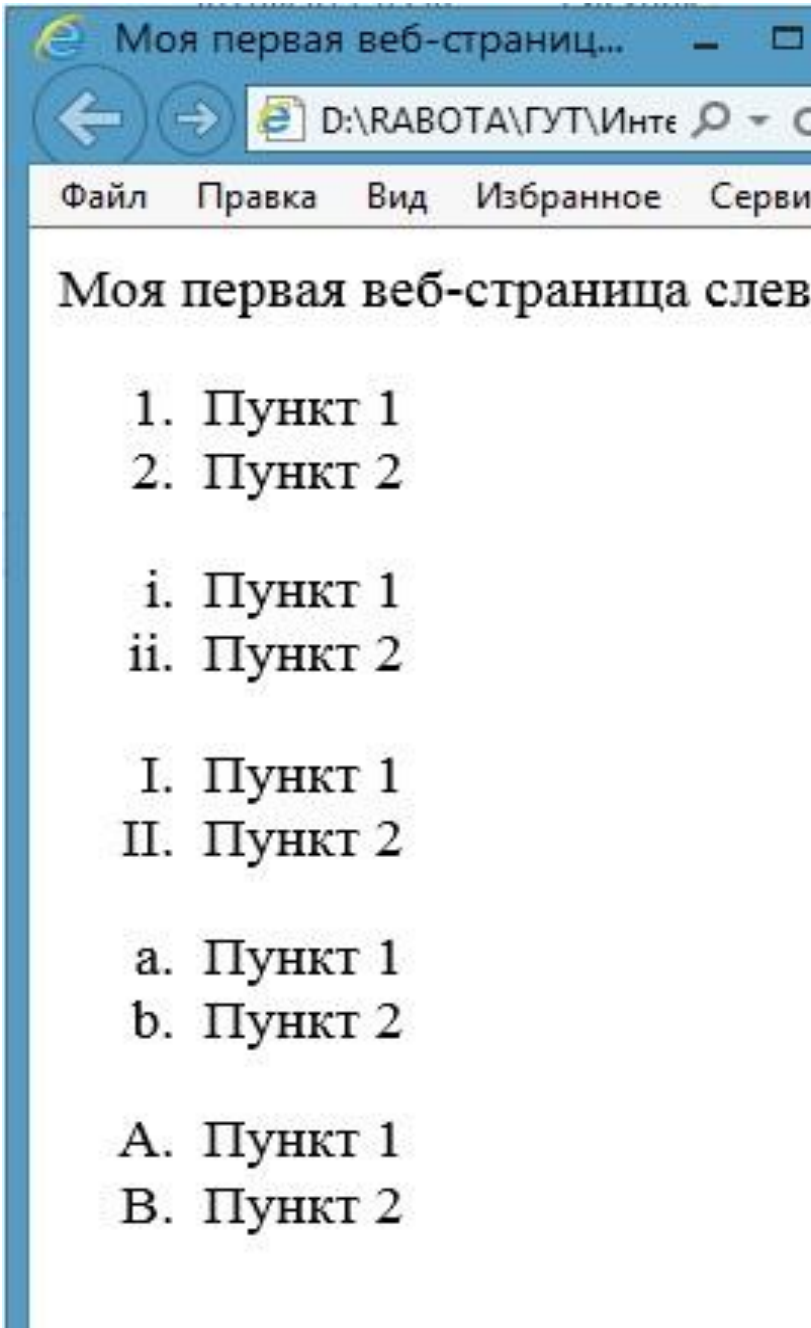


Нумерованные списки

- … - создание списка
- … - задание эл-в списка
- type – атрибуты списка
  - 1 – Арабские цифры
  - i – Строчные римские цифры
  - I – Прописные римские цифры
  - a – Строчные латинские буквы
  - A – Прописные латинские буквы

Пример нумерованного списка

```
<p align="left">Моя первая веб-страница слева</p>
<ol type="1">
  <li>Пункт 1</li>
  <li>Пункт 2</li>
</ol>
<ol type="i">
  <li>Пункт 1</li>
  <li>Пункт 2</li>
</ol>
<ol type="I">
  <li>Пункт 1</li>
  <li>Пункт 2</li>
</ol>
<ol type="a">
  <li>Пункт 1</li>
  <li>Пункт 2</li>
</ol>
<ol type="A">
  <li>Пункт 1</li>
  <li>Пункт 2</li>
</ol>
```



Списки определений

- <dl> … </dl> - создание списка
- <dt> … </dt> - тег понятия
- <dd> … </dd> - определение понятия

Пример списка определений

веб-страница слева
<dl>
<h3>Список определений</h3>
<dt>Электрическое напряжение между точками A и B электрической цепи или электрического поля</dt>
<dd>физическая величина, значение которой равно работе эффективного электрического поля (включающего сторонние поля), совершаемой при переносе единичного пробного электрического заряда из точки A в точку B</dd>
<dt>Электрическое сопротивление </dt>
<dd> физическая величина, характеризующая свойство проводника препятствовать прохождению электрического тока и равная отношению напряжения на концах проводника к силе тока, протекающего по нему</dd>
</dl>

веб-страница слева

Список определений

Электрическое напряжение между точками A и B электрической цепи или электрического поля

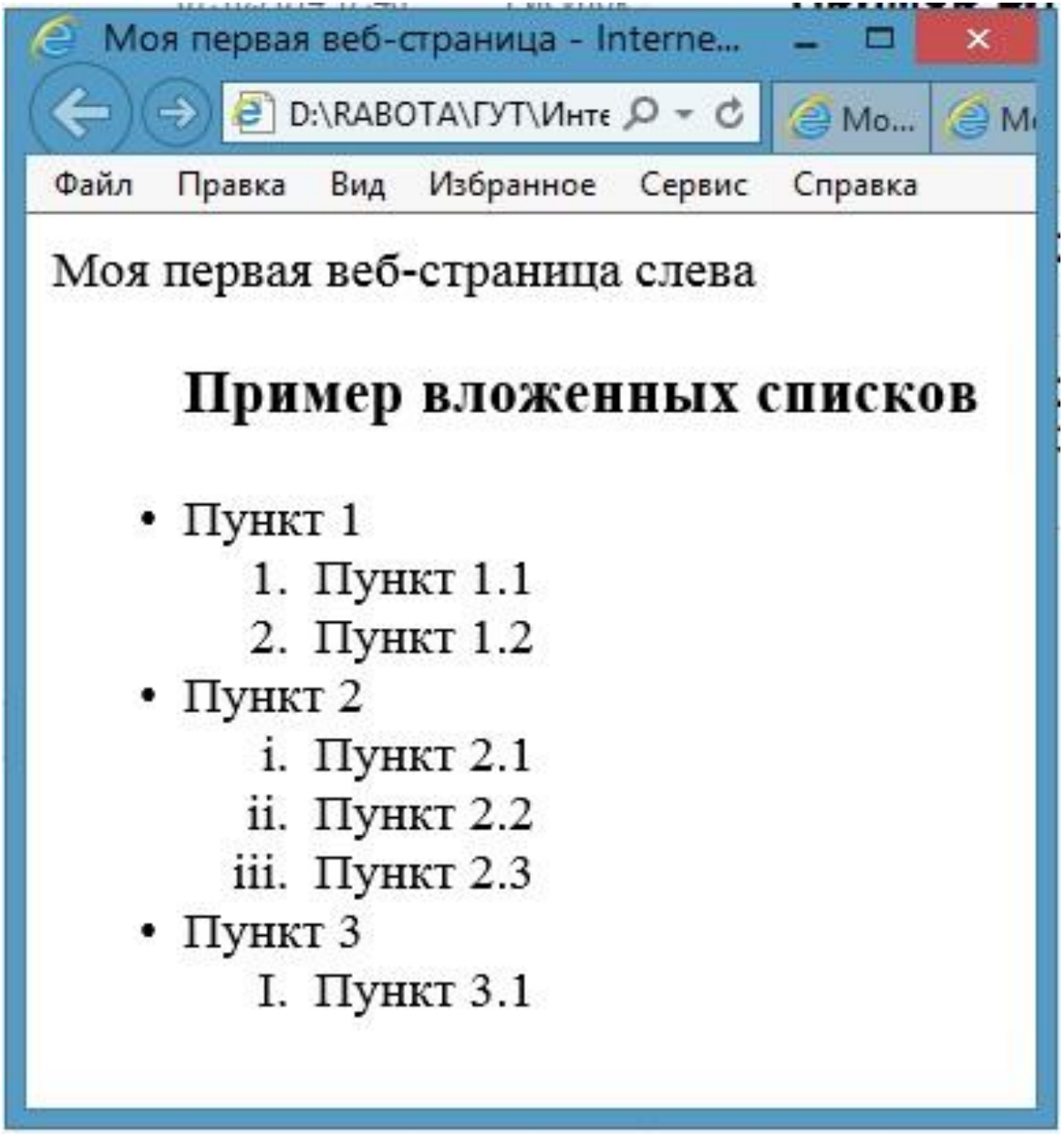
: физическая величина, значение которой равно работе эффективного электрического поля (включающего сторонние поля), совершаемой при переносе единичного пробного электрического заряда из точки A в точку B

Электрическое сопротивление

: физическая величина, характеризующая свойство проводника препятствовать прохождению электрического тока и равная отношению напряжения на концах проводника к силе тока, протекающего по нему

Вложенные списки

```
<ul>
  <h3>Пример вложенных списков</h3>
  <li>Пункт 1</li>
    <ol>
      <li>Пункт 1.1</li>
      <li>Пункт 1.2</li>
    </ol>
  <li>Пункт 2</li>
    <ol type="i">
      <li>Пункт 2.1</li>
      <li>Пункт 2.2</li>
      <li>Пункт 2.3</li>
    </ol>
  <li>Пункт 3</li>
    <ol type="I">
      <li>Пункт 3.1</li>
    </ol>
</ul>
```



Таблицы

- **<table>…</table> - создание таблицы**
- **<tr>…</tr> - создание строки**
- **<td>…</td> - создание столбца (ячейки)**
- **<th>…</th> - создание заголовка столбца (жирно и по центру)**
- **< caption>Заголовок таблицы< /caption>**

По умолчанию заголовки центрируются и размещаются либо над (<CAPTION ALIGN=top>), либо под таблицей (<CAPTION ALIGN=bottom>).

Атрибуты тега <table>

- **border="1" - толщина рамки**
- **bordercolor="#FF0000" – цвет рамки**
- **cellspacing="10" – отступ между ячейками**
- **cellpadding="10" - отступ внутри ячеек**
- **height="50px" – высота таблицы в пикселях**
- **width – ширина**
- **background="fon.jpg" – фон таблицы**
- **bgcolor (для TABLE, TR, TD) - цвет фона**

Атрибуты тегов <tr>, <td>, <th>

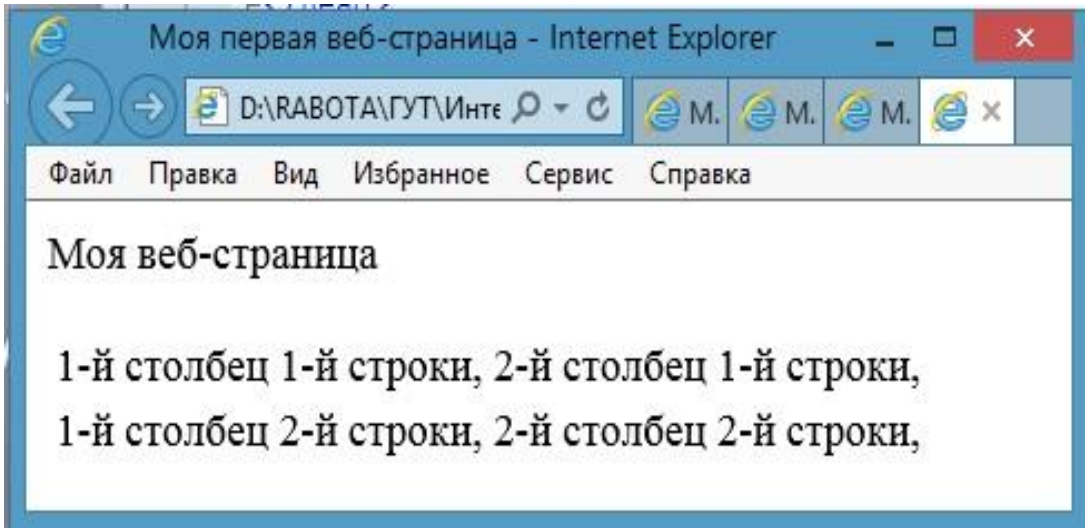
- **NOWRAP** (для <TH> или <TD>) - длина ячейки расширяется настолько, чтобы заключенный в ней текст поместился в одну строку.
- **ALIGN**
  - **bleedleft** прижимает содержимое ячейки вплотную к левому краю.
  - **left** выравнивает содержимое ячейки по левому краю с учетом отступа, заданного атрибутом CELLPADDING.
  - **center** располагает содержимое ячейки по центру.
  - **right** выравнивает содержимое ячейки по правому краю с учетом отступа, заданного атрибутом CELLPADDING.
- **VALIGN**
  - **top** выравнивает содержимое ячейки по ее верхней границе.
  - **middle** центрирует содержимое ячейки по вертикали.
  - **bottom** выравнивает содержимое ячейки по ее нижней границе.

Атрибуты тега <td>

- colspan = “2”- объединение столбцов
- rowspan = “3” - объединение строк
- width="75%" – ширина столбца от ширины таблицы
- height – высота строки
- bgcolor="#FF0000” – цвет ячейки
- background="fon.jpg" – фон ячейки

Пример таблицы 1

```
<body>
<p>Моя веб-страница</p>
<table>
  <tr>
    <td>1-й столбец 1-й строки,</td>
    <td>2-й столбец 1-й строки,</td>
  </tr>
  <tr>
    <td>1-й столбец 2-й строки,</td>
    <td>2-й столбец 2-й строки,</td>
  </tr>
</table>
</body>
```



Пример таблицы 2

```
<p>Моя веб-страница</p>
<table border="3" cellpadding="7" cellspacing="3" height="80"
width="50%">
  <caption>Пример простой таблицы</caption>
  <tr align="center">
    <td>1.1</td>
    <td>1.2</td>
    <td>1.3</td>
  </tr>
  <tr align="center">
    <td align="center">2.1</td>
    <td align="right">2.2</td>
    <td>2.3</td>
  </tr>
</table>
```

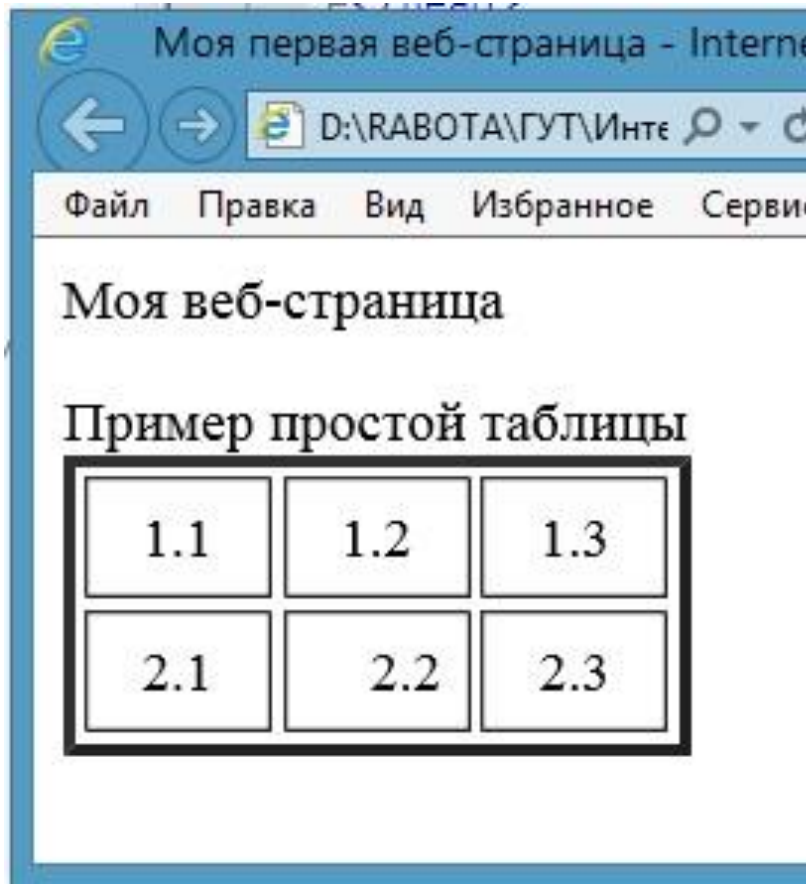
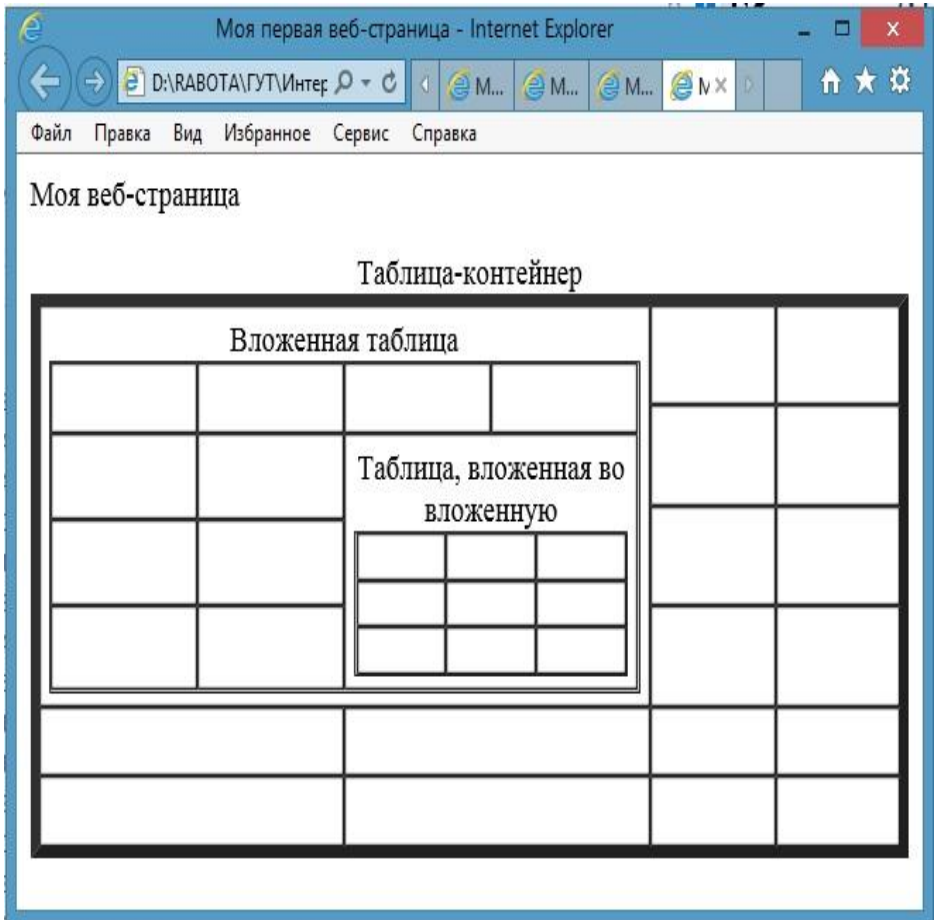


Таблица-контейнер

```
<table width="100%" border="5" cellspacing="0" cellpadding="5">
  <caption>Таблица-контейнер</caption>
  <tr>
    <td colspan="2" rowspan="4">
      <table border="1" cellspacing="0" cellpadding="5" width="100%">
        <caption>Вложенная таблица</caption>
        <tr>
          <td> </td>
          <td> </td>
          <td> </td>
          <td> </td>
        </tr>
        <tr>
          <td> </td>
          <td> </td>
          <td colspan="2" rowspan="3">
            <table width="100%" border="1" cellspacing="0" cellpadding="0">
              <caption>Таблица, вложенная во вложенную</caption>
              <tr>
                <td> </td>
                <td> </td>
                <td> </td>
              </tr>
              <tr>
                <td> </td>
                <td> </td>
                <td> </td>
              </tr>
              <tr>
                <td> </td>
                <td> </td>
                <td> </td>
              </tr>
            </table>
          </td>
        </tr>
<tr>
```



Гипертекстовые ссылки

- **<A>...</A>** - тег гиперссылки
- Атрибуты тега <A>:
  - **HREF** - адрес документа

    HREF = "http://www.sut.ru/index.html"
  - **NAME** - якорь - помеченное место в теле документа

ПРИМЕР

<A NAME="point"> - формирование якоря в документе

<A HREF="#point">Ссылка на якорь "point" в этом документе</A>

<A HREF="http://www.mysite.ru/index.html#point">Ссылка на точку "point" в документе "index.html" сайта "mysite.ru" </A>

Использование графики в HTML

- логотип компании на деловой странице;
- графика для рекламного объявления;
- различные рисунки;
- диаграммы и графики;
- художественные шрифты;
- подпись автора страницы;
- применение графической строки в качестве горизонтальной разделительной линии;
- применение графических маркеров для создания красивых маркированных списков.

Тег < IMG >

< IMG > - непарный тег вставки изображения

Атрибуты:

- **SRC** - имя графического файла (обязательный)
- **ALT** - подпись вместо рисунка (необязательный)
- **ALIGN** – выравнивание относительно окружающего текста

["top" | "middle"| "bottom"] ["left" | "right"]

- **BORDER** - рамка вокруг рисунка
- **HSPACE** - расстояние по горизонтали между рисунком и текстом
- **VSPACE** - ...по вертикали...
- **WIDTH и HEIGHT** - размеры изображения

<IMG SRC="image.gif" ALT="изображение" WIDTH="100" HEIGHT="200" HSPACE="10" VSPACE="10" BORDER="2" ALIGN="left">

ТАБЛИЦЫ CSS

Текст HTML со стилем

```
<!DOCTYPE HTML>
<html>
 <head>
   <title>О таблицах CSS</title>
   <meta charset="utf-8">
   <link rel="stylesheet" href="style.css">
 </head>
 <body>
    <h1>О таблицах CSS</h1>
    <p>
   Название "Каскадные таблицы стилей" происходит от английского Cascading Style Sheets, аббревиатурой которого является CSS.
       Впервые стили появляются в HTML 4.0 для определения представления элементов HTML и решения проблем представления документов.
       Стили обычно хранятся в таблицах стилей: могут быть определены как внутри HTML-документа, так и в специальном файле с расширением css.
       Используя отдельные файлы для хранения таблиц стилей, можно существенно сократить объем работы. Также можно определить несколько стилей, которые,
       подчиняясь существующим правилам, будут каскадно задавать один определенный стиль.
    </p>
  </body>
</html>
```

Пример стилевого файла style.css

```
body {
  font-family: Arial, Verdana,  sans-serif; /* Семейство шрифтов */
  font-size: 11pt; /* Размер основного шрифта в пунктах  */
  background-color: #f0f0f0; /* Цвет фона веб-страницы */
  color: #333; /* Цвет основного текста */
}
h1 {
  color: #a52a2a; /* Цвет заголовка */
  font-size: 24pt; /* Размер шрифта в пунктах */
  font-family: Georgia, Times, serif; /* Семейство шрифтов */
  font-weight: normal; /* Нормальное начертание текста  */
}
p {
  text-align: justify; /* Выравнивание по ширине */
  margin-left: 60px; /* Отступ слева в пикселах */
  margin-right: 10px; /* Отступ справа в пикселах */
  border-left: 1px solid #999; /* Параметры линии слева */
  border-bottom: 1px solid #999; /* Параметры линии снизу */
  padding-left: 10px; /* Отступ от линии слева до текста  */
  padding-bottom: 10px; /* Отступ от линии снизу до текста  */
}
```

Типы стилей

Различают несколько типов стилей, которые могут совместно применяться к одному документу. Это:

- **стиль браузера**
- **стиль пользователя**
- **стиль автора**

Стиль браузера

Оформление, которое по умолчанию применяется к элементам веб-страницы браузером.

Это оформление можно увидеть в случае «голого» HTML, когда к документу не добавляется никаких стилей.

Например, заголовок страницы, формируемый тегом <H1>, в большинстве браузеров выводится шрифтом с засечками размером 24 пункта.

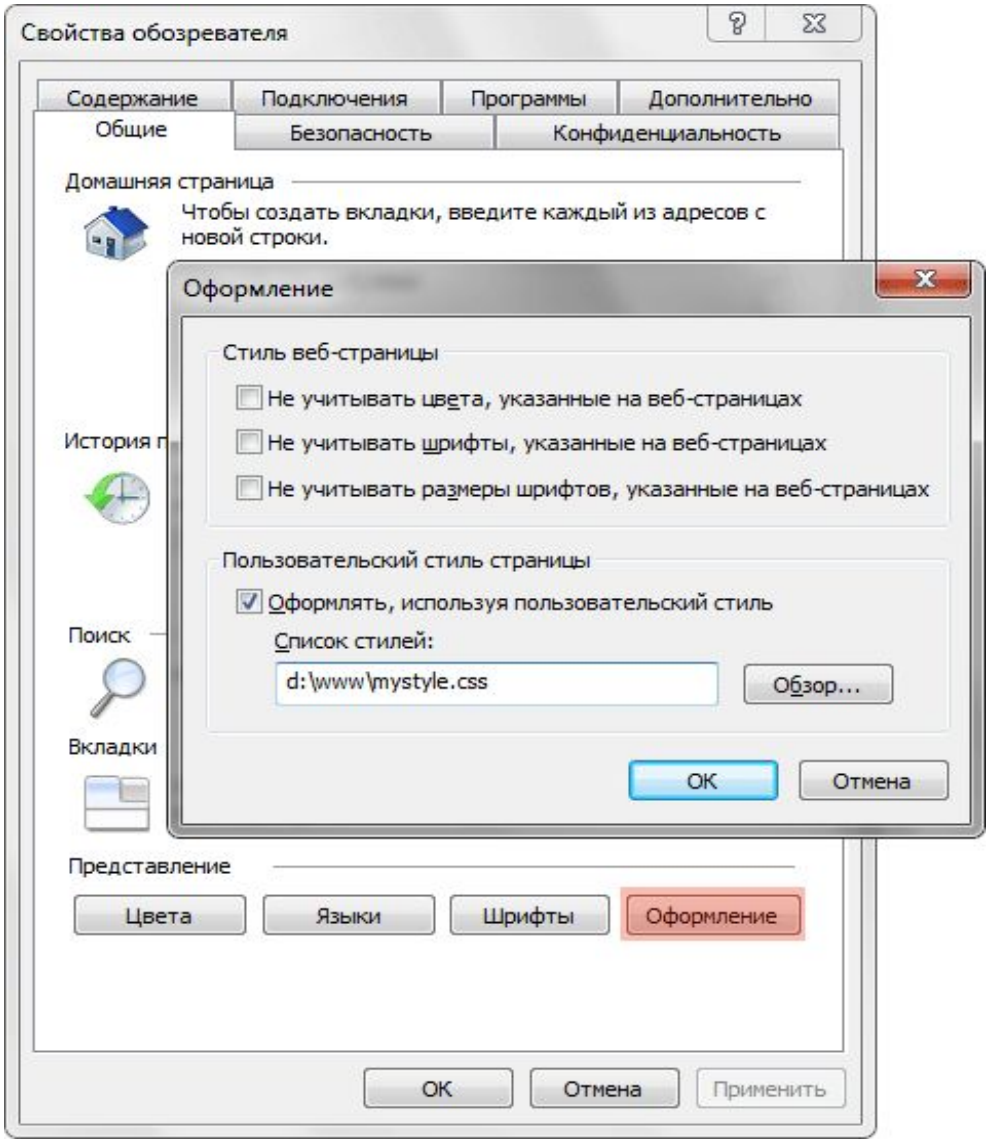
Стиль пользователя

Это стиль, который может включить пользователь сайта через настройки браузера.

Такой стиль имеет более высокий приоритет и переопределяет исходное оформление документа.

В браузере Internet Explorer подключение стиля пользователя делается через меню Сервис>Свойство обозревателя> Кнопка «Оформление»

Стиль пользователя (Internet Explorer)



Стили автора (программиста)

- Внутренний – для одного тега
- Глобальный – для одной страницы
- Внешний (связанный) – для многих

Приоритеты стилей

Для каскадных таблиц стилей определен приоритет использования:

- стили, используемые по умолчанию браузером;
- стили, хранящиеся во внешней таблице;
- стили, хранящиеся во внутренней таблице стилей (внутри тега <head>);
- встроенный стиль (внутри элемента HTML).

Синтаксис CSS

***Синтаксис CSS* состоит из трех частей:**

- **селектора**
- **свойства**
- **значения:**

селектор {свойство: значение}

Виды селекторов

- **Стиль** для конкретного элемента (тега).

  Применяться будет для всех конкретных элементов.

- **Класс**

Класс используется для одного или нескольких элементов, которые присутствуют на странице: параграфов, ссылок, кнопок, полей ввода.

  - **Для конкретного элемента** (тега).

    Применяться будет для конкретных элементов при особом указании.

  - **Универсальный.**

    Применяться будет для любых элементов при особом указании.

- **Идентификатор**

  - Значение идентификатора может быть присвоено только одному элементу **HTML. ID** используется для отдельных элементов, которые присутствуют на странице только в одном месте.
  - Приоритет выше, чем у класса.

Правила для свойств

- Свойство и значение разделяются двоеточием и помещаются внутри фигурных скобок:

p {font-size: 75%}

- Если значение состоит из нескольких слов, то необходимо поместить значение в кавычки:

h1 {font-family: "lucida calligraphy"}

- Если требуется определить более одного свойства, то необходимо разделить свойства точкой с запятой:

table { font-family: arial, "sans serif"; border-style: dotted}

Формы записи стиля

- Задание свойств по отдельности

td { background: olive; }
td { color: white; }
td { border: 1px solid black; }

- Задание свойств группой.

Если требуется определить более одного свойства, то необходимо разделить свойства точкой с запятой:

table { font-family: arial, "sans serif"; border-style: dotted}

```
td {
  background: olive;
  color: white;
  border: 1px solid black;
}
```

Комментарии

```
p {
  width: 200px; /* Ширина блока */
  margin: 10px; /* Поля  вокруг элемента */
  float: left; /* Обтекание по правому краю */
}
```

Способы добавления стилей

- Связанные стили
- Глобальные стили
- Внутренние стили

Связанные (внешние) стили

- Описание стилей в отдельном файле *.css
- Область действия – все документы, которые подключают этот стилевой файл.
- Для связывания используется тег **<link>** в контейнере <head>
- Атрибуты тега <link>:
  - rel – отношение между документом и файлом
  - charset – кодировка связываемого документа
  - href – путь к связываемому файлу

Пример подключения связанного типа

```
<head>
   <meta charset="utf-8">
   <title>Стили</title>
   <link rel="stylesheet" href="mysite.css">
   <link rel="stylesheet" href="main.css">
 </head>
 <body>
    <h1>Заголовок</h1>
    <p>Текст</p>
 </body>
```

Пример файла со стилем

```
h1 {
  color: #000080;
  font-size: 200%;
  font-family: Arial, Verdana, sans-serif;
  text-align: center;
}

p {
  padding-left: 20px;
}
```

Семейства шрифтов

- **serif** — шрифты с засечками (антиквенные), типа Times;
- **sans-serif** — рубленные шрифты (шрифты без засечек или гротески), типичный представитель — Arial;
- **cursive** — курсивные шрифты;
- **fantasy** — декоративные шрифты;
- **monospace** — моноширинные шрифты, ширина каждого символа в таком семействе одинакова (шрифт Courier)

Глобальные стили

- Свойства CSS описываются в самом документе и располагаются в заголовке веб-страницы

- Область действия – весь текущий документ.

Пример глобального стиля

```
<head>
   <meta charset="utf-8">
   <title>Глобальные стили</title>
   <style>
      h1 {
         font-size: 120%;
         font-family: Verdana, Arial, Helvetica, sans-serif;
         color: #333366;
       }
   </style>
</head>
<body>
    <h1>Это текст со стилем</h1>
</body>
```

Внутренние стили

- Внутренний или встроенный стиль - расширение для одиночного тега на текущей веб-странице.
- Больше возможностей, чем у атрибутов тега.
- Для определения стиля используется атрибут **style**
- Значением атрибута style выступает набор стилевых правил
- (-) Большой объем файла документа и время его загрузки
- (-) Сложность редактирования

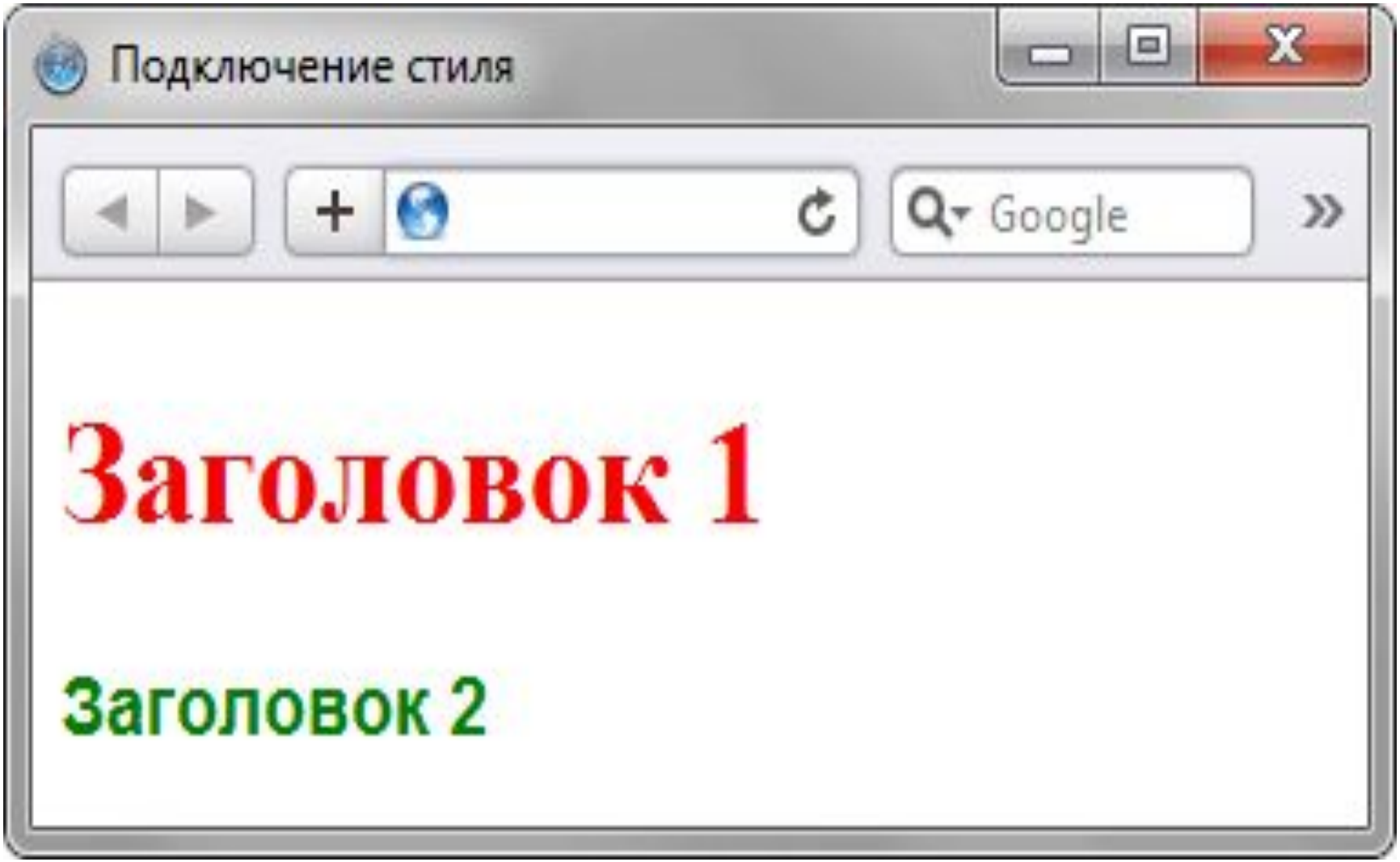
Пример внутреннего стиля

```
<head>
    <meta charset="utf-8">
    <title>Внутренние стили</title>
 </head>


 <body>
     <p style="font-size: 120%; font-family:
monospace; color: #cd66cc">Пример текста</p>
</body>
```

Сочетание разных методов

```
<head>
  <meta charset="utf-8">
  <title>Подключение стиля</title>
  <style>
     h1 {
       font-size: 120%;
       font-family: Arial, Helvetica, sans-serif;
       color: green;
     }
  </style>
 </head>

 <body>
     <h1 style="font-size: 36px; font-family: Times, serif; color: red">Заголовок 1</h1>
     <h1>Заголовок 2</h1>
 </body>
```

Пример сочетания разных методов



Какой цвет у текста?

```
<head>
  <title>Цвет текста</title>
  <style>
    HTML { color: black; }
    BODY { color: red; }
    P { color: green; }
  </style>
 </head>

<body>
  <p style="color: blue;"><span style="color:
olive;">Текст</span></p>
 </body>
```

Классы

Классы применяют, когда необходимо определить стиль для индивидуального элемента веб-страницы или задать разные стили для одного тега.

- Описание класса:

Тег.Имя класса { свойство1: значение; свойство2: значение; ... }

- Применение класса в коде HTML:

к тегу добавляется атрибут

class="Имя класса"

Пример использования классов

```
<head>
  <style>
    p { /* Обычный абзац */
       text-align: justify; /* Выравнивание текста по ширине */
       color: red; /* Цвет текста */
    }
    p.cite { /* Абзац с классом cite */
       color: navy; /* Цвет текста */
       margin-left: 20px; /* Отступ слева */
       border-left: 1px solid navy; /* Граница слева от текста */
       padding-left: 15px; /* Расстояние от линии до текста */
    }
  </style>
</head>
<body>
```

Задачами изучения дисциплины является реализация требований, установленных Государственными образовательными стандартами высшего профессионального образования к уровню подготовки специалистов в проектирования вычислительных машин и сетей.

 В ходе изучения дисциплины ставится задача сформировать взгляд на проектирование и эксплуатацию вычислительных машин и сетей как на систематическую научно-практическую деятельность, носящую прикладной характер.

```
</body>
```

Пример использования классов

Задачами изучения дисциплины является реализация требований, установленных Государственными образовательными стандартами высшего профессионального образования к уровню подготовки специалистов в проектирования вычислительных машин и сетей.

> В ходе изучения дисциплины ставится задача сформировать взгляд на проектирование и эксплуатацию вычислительных машин и сетей как на систематическую научно-практическую деятельность, носящую прикладной характер.

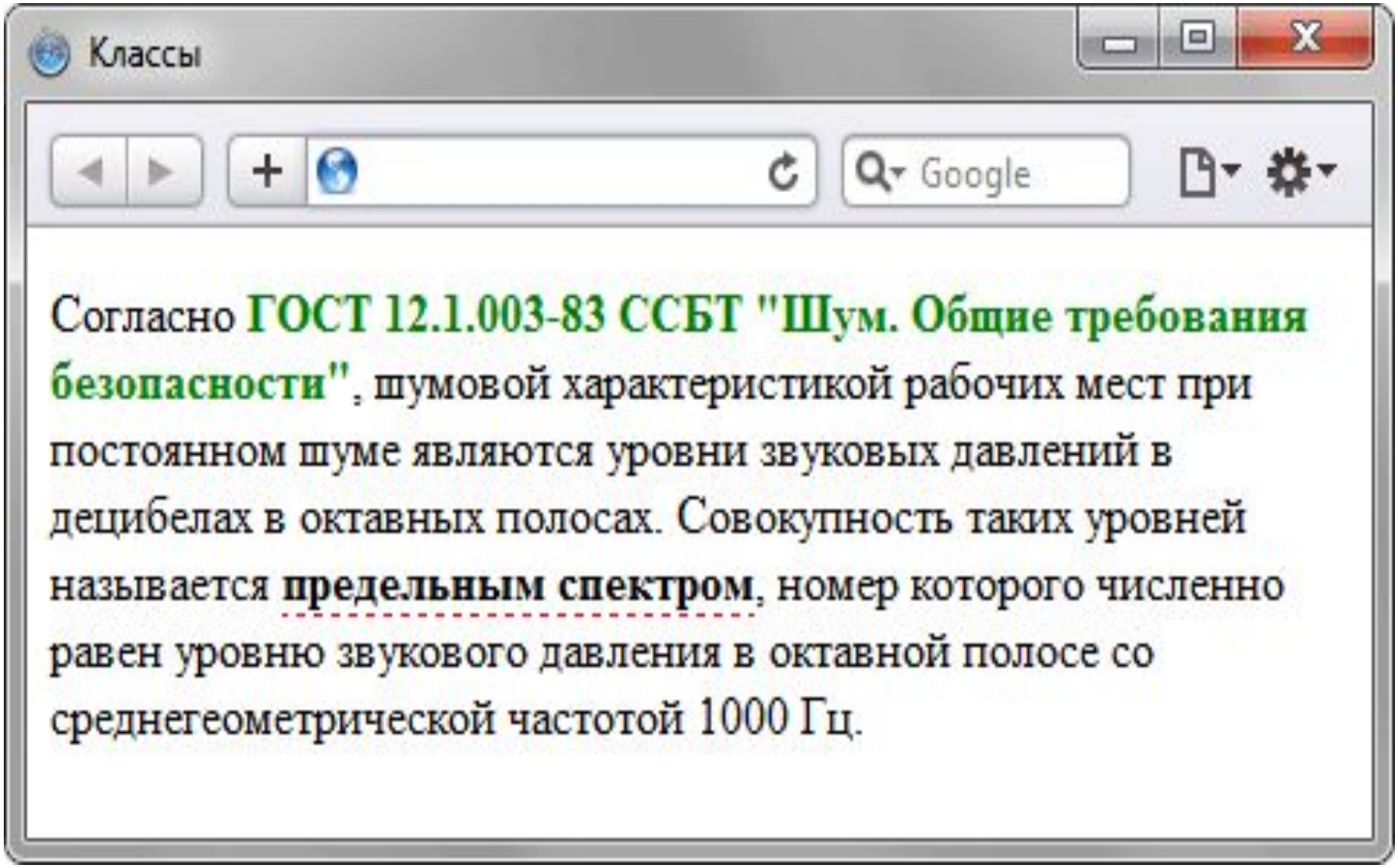
Класс без указания тега

Класс без указания тега можно использовать для любого тега, к которому применимы описанные в классе свойства

```
<style>
  .gost {
   color: green; /* Цвет текста */
   font-weight: bold; /* Жирное начертание */
  }
  .term {
   border-bottom: 1px dashed red; /* Подчеркивание под текстом */
  }
 </style>
```

```
<p>Согласно <span class="gost">ГОСТ 12.1.003-83 ССБТ &amp;quot;Шум. Общие  требования безопасности&amp;quot;</span>, шумовой характеристикой рабочих  мест при постоянном шуме являются уровни звуковых давлений в децибелах  в октавных полосах. Совокупность таких уровней называется <b class="term">предельным спектром</b>, номер которого численно равен уровню звукового давления в октавной полосе со среднегеометрической частотой 1000&amp;nbsp;Гц. </p>
```

Пример класса без тега

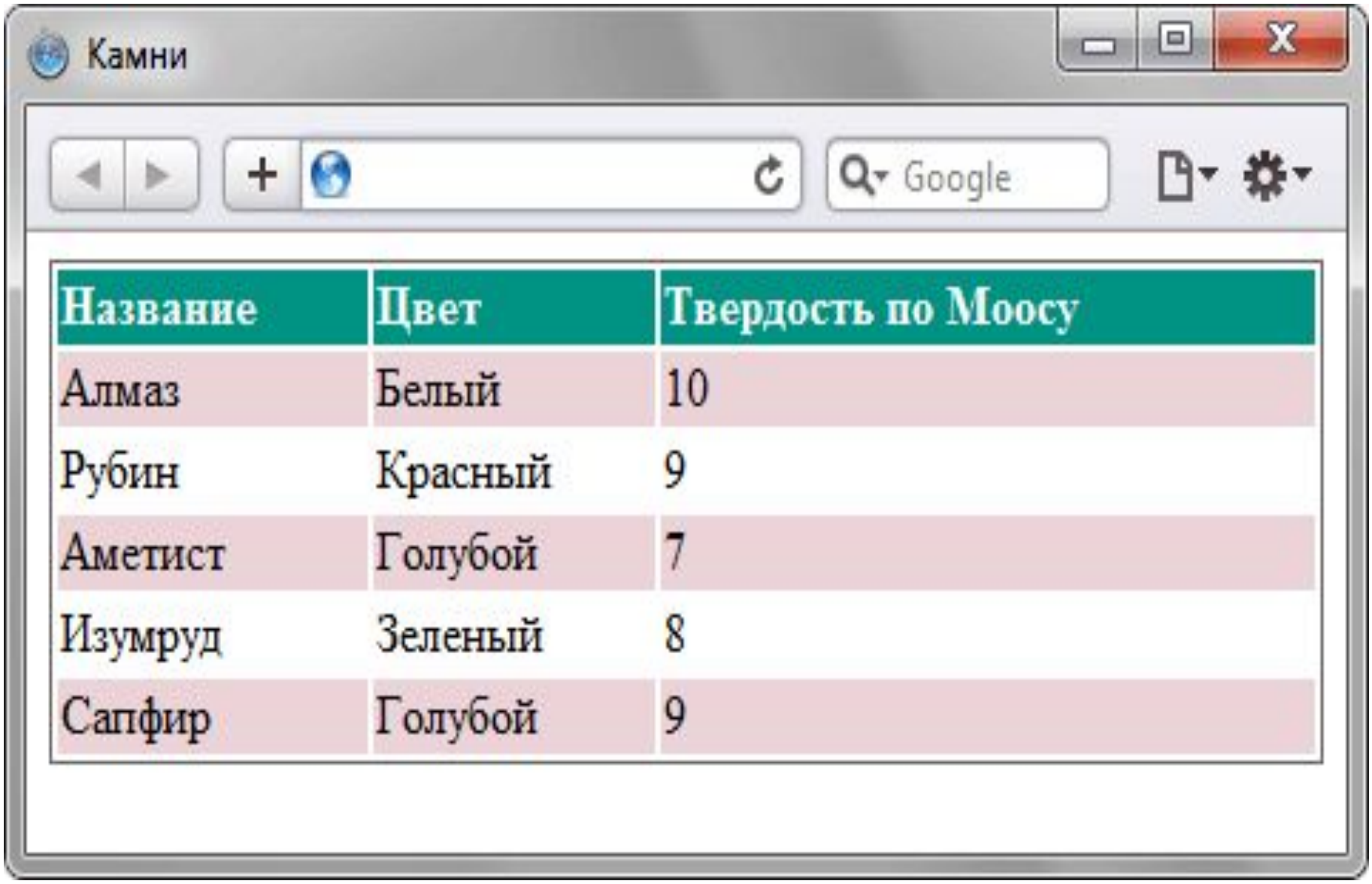


Пример таблицы-зебры

```
<head>
  <title>Камни</title>
  <style>
    table.jewel {
      width: 100%; /* Ширина таблицы */
      border: 1px solid #665; /* Рамка вокруг таблицы */
    }
    th {
      background: #009383; /* Цвет фона */
      color: #fff; /* Цвет текста */
      text-align: left; /* Выравнивание по левому краю */
    }
    tr.odd {
      background: #ebd3d7; /* Цвет фона */
    }
  </style>
</head>
```

```
<body>
  <table class="jewel">
    <tr>
      <th>Название</th><th>Цвет</th><th>Твердость по Моосу</th>
    </tr>
    <tr class="odd">
      <td>Алмаз</td><td>Белый</td><td>10</td>
    </tr>
    <tr>
      <td>Рубин</td><td>Красный</td><td>9</td>
    </tr>
    <tr class="odd">
      <td>Аметист</td><td>Голубой</td><td>7</td>
    </tr>
    <tr>
      <td>Изумруд</td><td>Зеленый</td><td>8</td>
    </tr>
    <tr class="odd">
      <td>Сапфир</td><td>Голубой</td><td>9</td>
    </tr>
  </table>
</body>
```

Таблица-зебра



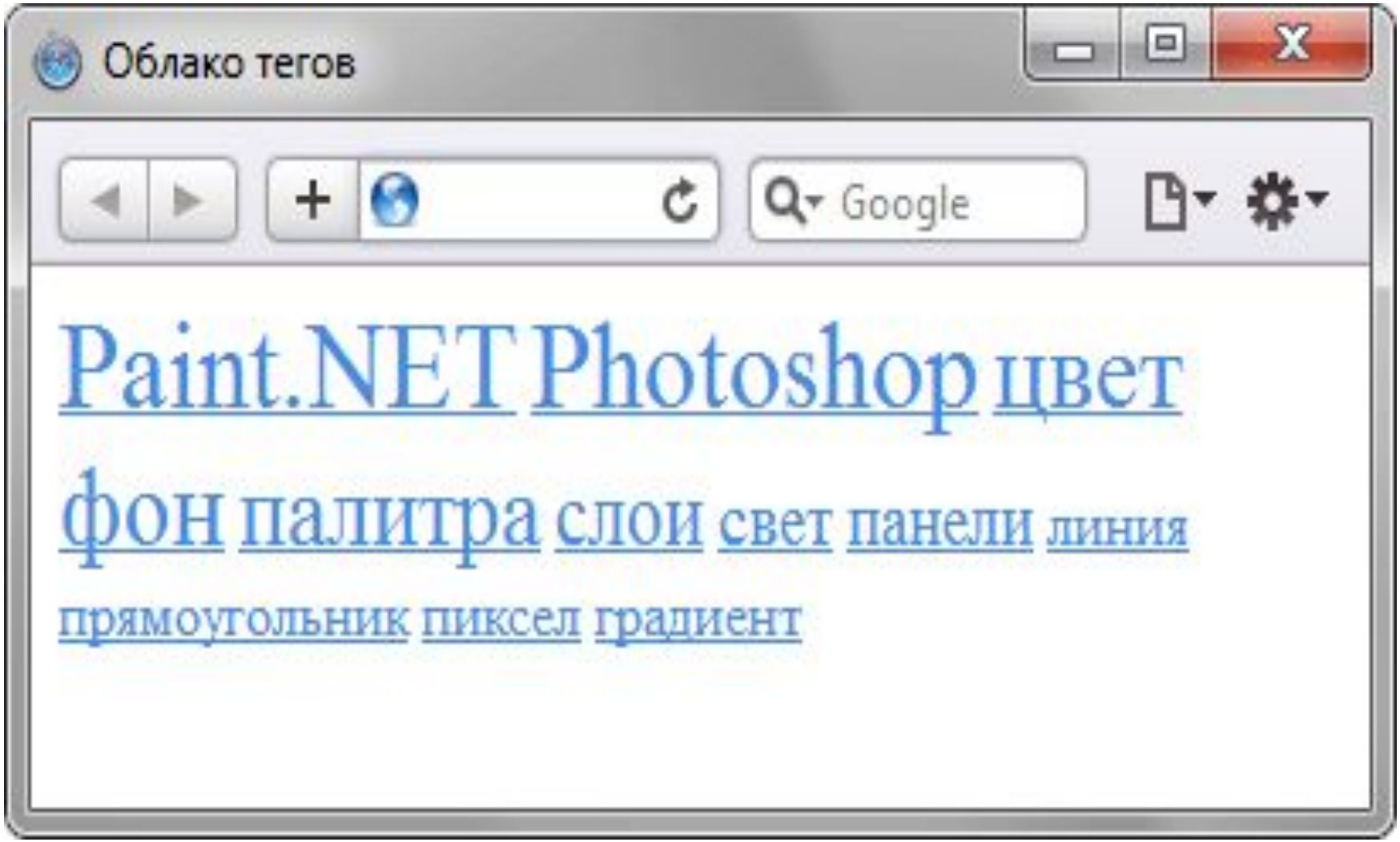
Название	Цвет	Твердость по Моосу
Алмаз	Белый	10
Рубин	Красный	9
Аметист	Голубой	7
Изумруд	Зеленый	8
Сапфир	Голубой	9

Одновременное использование разных классов

```
<head>
  <title>Облако тегов</title>
  <style type="text/css">
    .level1 { font-size: 1em; }
    .level2 { font-size: 1.2em; }
    .level3 { font-size: 1.4em; }
    .level4 { font-size: 1.6em; }
    .level5 { font-size: 1.8em; }
    .level6 { font-size: 2em; }
    a.tag { color: #468be1; /* Цвет ссылок */ }
  </style>
</head>
```

```
<body>
  <div>
    <a href="/term/2"   class="tag level6">Paint.NET</a>
    <a href="/term/69" class="tag level6">Photoshop</a>
    <a href="/term/3"   class="tag level5">цвет</a>
    <a href="/term/95" class="tag level5">фон</a>
    <a href="/term/11" class="tag level4">палитра</a>
    <a href="/term/43" class="tag level3">слои</a>
    <a href="/term/97" class="tag level2">свет</a>
    <a href="/term/44" class="tag level2">панели</a>
    <a href="/term/16" class="tag level1">линия</a>
    <a href="/term/33" class="tag level1">прямоугольник</a>
    <a href="/term/14" class="tag level1">пиксел</a>
    <a href="/term/27" class="tag level1">градиент</a>
  </div>
 </body>
```

Пример сочетания классов



ПРОГРАММИРОВАНИЕ НА JAVASCRIPT

Виды скриптов

- **Внутренние**

(внутри HTML-документа)

- **Внешние**

(в отдельном файле)

Структура программы

```
<script language = "JavaScript">

    ТЕЛО СКРИПТА

</script>
```

Комментарии

- // Текст однострочного комментария после двойной наклонной черты
- /*Текст многострочного комментария размещен между двумя конструкциями*/

Пример однострочного комментария

```
// Этот комментарий занимает всю строку
alert(‘Всем привет!');

alert(‘Пишите письма!'); // Этот комментарий следует за инструкцией
```

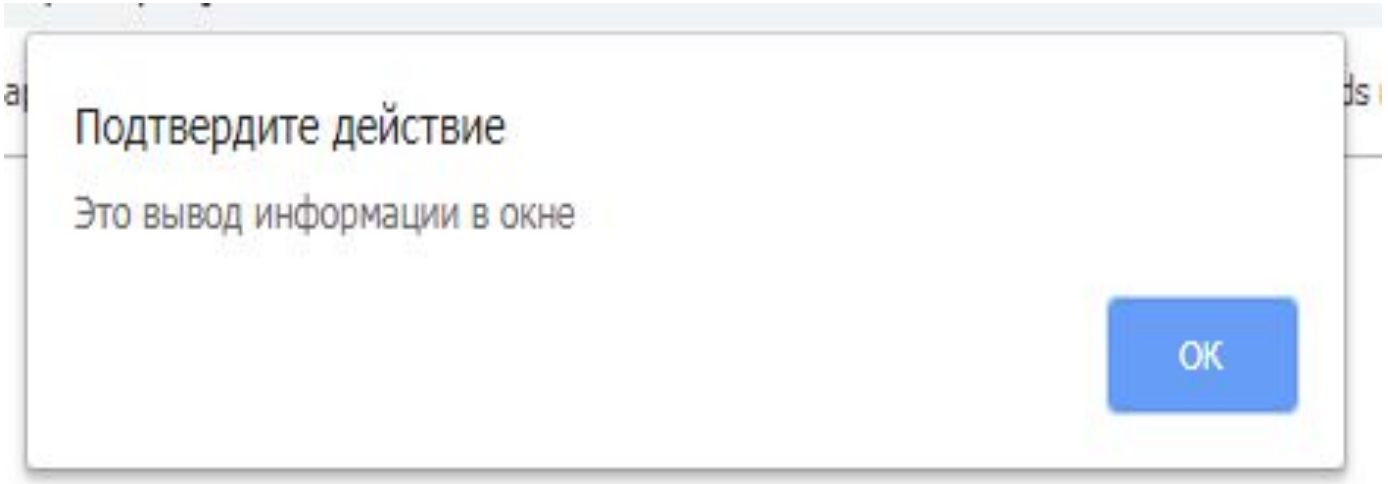
Пример многострочного комментария 1

```
/* Пример с двумя сообщениями. Это - многострочный комментарий. */
alert(‘Всем привет!');
alert(‘Пишите письма');
```

Пример многострочного комментария 2

```
/* Пример с двумя сообщениями. Это - многострочный комментарий.
alert(‘Всем привет!'); */
alert(‘Пишите письма');
```

Скрипт внутри HTML-документа




```
<html>
<head>
…
</head>

<body>
   <p>Начало документа...</p>
   <script>
      alert( 'Это вывод информации в окне' );
   </script>
   <p>...Конец документа</p>
 </body>
</html>
```

Алгоритм исполнения скрипта

Браузер

1. Начинает отображать страницу, показывает часть документа до script.
2. Встретив тег script, переключается в JavaScript-режим и исполняет его содержимое.
3. Закончив выполнение, возвращается обратно в HTML-режим и *только тогда* отображает оставшуюся часть документа.

Внешний скрипт

- Сам скрипт размещается в отдельном файле *.js
- Скрипт подключается к HTML-документу при помощи тега <script> в том месте html-документа, где должен быть исполнен

<script src=“myscript.js''></script>

- В одном теге SCRIPT нельзя одновременно подключить внешний скрипт и указать код. Код не будет исполнен. Надо по отдельности два тега.

Ассинхронное выполнение

- Скрипты могут выполняться долго и тормозить отображение html-контента.
- Для ассинхронного исполнения есть атрибуты
  - async
  - defer
- **async** – браузер (кроме IE9) при обнаружении скрипта не останавливает выполнение последующего html-кода. Если несколько скриптов, то исполнен будет тот, кто первым загрузится. Пример: <script async src="...">
- **defer** – относительный порядок скриптов сохраняется и они будут исполнены только после полной загрузки html-кода
- Атрибуты только для внешних (!) скриптов.

Типы данных

- **Численные**
  - **Целые**
  - **Вещественные**
- **Булевские**
- **Строковые**
- **Объектные**

Задание типов данных

Переменные

var (устарело)

let (современный вариант)

- **var a = 35;**
- **let a = 37.3;**
- **let a**

  a = “Строка текста”;

  let b = true;

  c = false;

Регистр в именах – значим.

Задание типов данных

Константы

```
const MyString = ‘Пример моей строки’;
const COLOR_RED = "#F00";
const COLOR_GREEN = "#0F0";
const COLOR_BLUE = "#00F";
const COLOR_ORANGE = "#FF7F00";

let color = COLOR_GREEN; // Так можно
MyString = ‘Новое значение’; // Так нельзя!!!
```

Типы данных

Числа (number):

- **Целые или вещественные**

```
let n = 15; // целая переменная
let x = 1.618; // вещественная переменная
n = 12.3; // теперь это вещественная переменная
```

- **Бесконечность (Infinity)**

```
alert(5/0);// Infinity
```

- **Вычислительная ошибка (NaN)**

```
alert(“”/5); // NaN
```

Типы данных

Строки (string)

```
let str1 = "Текст в двойных кавычках";
let str2 = 'Текст в одинарных кавычках';
let str3 = `Обратные кавычки позволяют встраивать переменные ${str1}`;
```

Выражение внутри ${…} вычисляется, и его результат становится частью строки.

Типы данных

Булевские (логические) данные (boolean)

Только два значения: true (истина) и false (ложь).

```
let bool1 = true;
let bool2 = false;
let bool3 = 7>4;
alert(bool3);
```

Типы данных

Неопределенные типы

- **null** (пусто, значение неизвестно)

```
let MyAge = null;
```

- **undefined** (значение не было присвоено)

```
let y;
let z;
z= undefined;
alert(y); // undefined
alert(z); // undefined
```

Типы данных

- **Объекты** (object)

  Для хранения коллекций данных или более сложных объектов.

- **Символы** (symbol)

  Для создания уникальных идентификаторов объектов.

Операторы

- ***Операнд*** – то, к чему применяется оператор.

```
y = z - x;
```

- ***Унарный*** **оператор** – оператор, который применяется к одному операнду.

```
y = - x;
```

- ***Бинарный*** **оператор** – оператор, который применяется к двум операндам.

```
y = z - x;
```

Операции

- **С числами:**
  - **Сложение +**
  - **Вычитание -**
  - **Умножение ***
  - **Деление /**
- **Со строками:**
  - **Конкатенация (объединение)**

Var a = “a” + “b” + “c”;

Бинарный +

- Сложение чисел

```
z = x + y;
```

- Объединение строк

```
s = ‘abc’ + ‘xyz’;
```

если хотя бы один операнд является строкой, то второй будет также преобразован к строке

```
alert( ‘3' + 7 ); // “37“
alert(3 + 3 + ‘7' ); // “67", а не “337“
alert( 7 - ‘3' ); // 4
```

Унарный ‘+’

- С числом ничего не делает

```
alert(+5); // 5
```

- Строку преобразует в число

```
alert(+”5”); // 5
alert(+true); // 1
let a=“3”;
let b=“7”;
alert(a + +b); // 10
```

Приоритеты операторов

Приоритет – порядок выполнения операторов в выражении.

У унарных операторов приоритет выше, чем у соответствующих бинарных

Приоритет	Название	Обозначение
max	**унарный плюс**	**+**
max	**унарный минус**	**-**
	умножение	*****
	деление	**/**
	сложение	**+**
	вычитание	**-**
	…	**…**
min	**присваивание**	**=**

Присваивание

```
let a = 3*7 + 16;
alert(a); // 37

//присваивание по цепочке
let x, y, z;
x = y = z = 3 + 7; //

// присваивание как часть сложного выражения
let a = 3;
let b = 7;
let c = 3 - (a = b + 1); // -5
```

Остаток от деления %

x = 7 % 3; // 1 – остаток от деления 7 на 3

Возведение в степень **

```
x = 5**2; // 25
y = 2**3; // 8
z = x**y; // 25 в степени 8


//дробная степень
c = 81**(1/2);// 9
d = 81**(1/4);// 3
```

Инкремент/Декремент

- Применимо **только к переменным**
- **Инкремент** (увеличение операнда на 1)

```
let counter = 0;
counter++;//  или ++counter
alert(counter); // 1
```

- **Декремент** (уменьшение операнда на 1)

```
let counter = 10;
counter--;// или --counter
alert(counter); // 9
```

Операторы ++/-- могут также использоваться внутри выражений. Их приоритет выше, чем у арифметических операций.

Префиксная и постфиксная формы Инкремента/Декремента

- **Префиксная** форма (операция ++ или -- перед операндом)

```
let counter = 10;
let x = ++counter; // префиксная форма
alert(x); // 11
```

- **Постфиксная** форма (операция после)

```
let counter = 10;
let x = counter++; // префиксная форма
alert(x); // 10
```

Побитовые операторы

Для целых 32-битных чисел.

Сами операторы работают над двоичным представлением чисел

- AND(и) (&)
- OR(или) (|)
- XOR(побитовое исключающее или) (^) только у одного есть 1.
- NOT(не) (~)

Сокращенная арифметика

```
// полная запись
let x = 3;
x = x + 7;
x = x * 5;
alert( x ); // 50

// сокращенная запись
let x = 3;
x += 7; // (работает как x = x + 7)
x *= 5; // (работает как x = x * 5)
alert( n ); // 50
```

Операторы сравнения

== - равенство с приведением типов

!= - неравенство

< - меньше

> - больше

<= - меньше или равно

>= - больше или равно

=== - равенство строгое (без приведения типов)

!== - неравенство строгое

результат сравнения имеет булевский тип

Примеры операторов сравнения

```
alert( 7 > 3 ); // true (верно)
alert( 3 == 7 ); // false (неверно)
alert( 7 != 3 ); // true (верно)

let a = (7>3);
alert(a);// true (верно)
```

Сравнение строк

Строки сравниваются посимвольно в алфавитном порядке

Алгоритм сравнения строк:

- Сначала сравниваются первые символы строк.
- Если первый символ первой строки больше (меньше) (по позиции в алфавите), чем первый символ второй строки, то первая строка больше (меньше) второй.
- Если первые символы равны, то таким же образом сравниваются уже вторые символы строк.
- Сравнение продолжается, пока не закончится одна из строк.
- Если обе строки заканчиваются одновременно, то они равны. Иначе, большей считается более длинная строка.

Примеры сравнения строк

alert(‘Ю' > ‘Б'); // true

alert(‘Торт' > ‘Торг'); // true

alert(‘Волынкин' > ‘Вол'); // true

Используется кодировка Unicode

Таблица кодов Unicode

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000		┌	┐	└	┘	│	─	•	◘						♫	♪
001	┼	◄	↕	‼	¶	┴	┬	┤	↑	├	→	←	└	↔	▲	▼
002		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
003	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
004	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
005	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
006	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
007	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

Сравнение разных типов

// строки приводятся к числам

alert(‘7' > 3); // true, строка ‘7' становится числом 7

alert('09' == 9); // true, строка '01' становится числом 1

// логические типы приводятся к числам

alert(true == 1); // true

alert(false == 0); // true

Строгое сравнение

Оператор строгого равенства === проверяет равенство без приведения типов

Если операнды имеют разные типы, то проверка === возвращает false без попытки их преобразования

```
alert( 1 === true ); // false
```

```
alert( null == undefined ); // true
alert( null === undefined ); // false
```

Задачки

- **5 > 4**
- **"арбуз" > "тыква"**
- **"7" > "37«**
- **"102" > “0101"**

Составной оператор

{…}

```
{
    y = 5*x**2 + 3*x + 7;
    z = 3*x**2 + 7*x + 12;
    alert(z>y);
}
```

Массивы

- **Одномерные**

```
var   a = new Array();
a[0] = 5;
a[1] = 2.5;
a[2] = “Строка”;
```

- **Многомерные**

```
var a = new Array();
a[0] = new Array();
a[0][0] = "1,1";
a[0][1] = "1,2";
a[1] = new Array();
a[1][0] = "2,1";
a[1][1] = "2,2";
```

Ввод-вывод данных

- **Ввод данных** через модальное окно

  имя_переменной = **prompt**(msg, defaultText);

```
var a = prompt("Input A: ", 10);
var b = prompt("Input B: ", "");
a = parseFloat(a);
b = parseFloat(b);
```

- **Ввод булевских данных через модальное окно**

  confirm(вопрос);

выводит окно с вопросом и с двумя кнопками: OK и CANCEL. Результатом будет true при нажатии OK и false – при CANCEL(Esc).

- **Вывод данных через модальное окно**

  alert(text);

- **Вывод данных в HTML-документ**

  document.write(text);

Условные операторы

- **if (условие) { …} else { …}**

- **?**

- **Switch(число)**

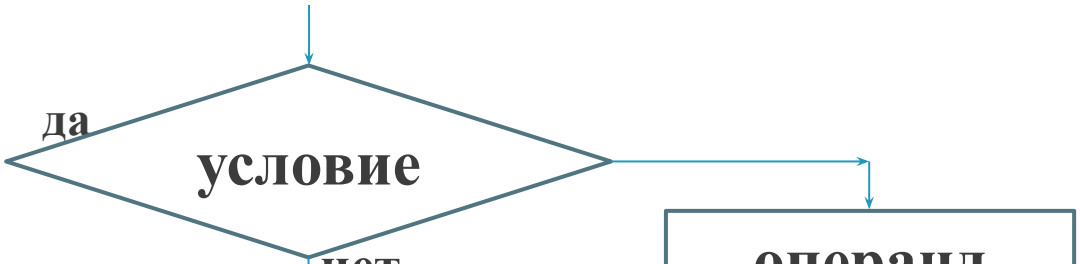
```
{
   case вариант-1: {…} break;
   case вариант-2: {…} break;
   case вариант-3: {…} break;
   default: {…} break;
}
```

  - Если break нет, то выполнение пойдёт ниже по следующим case.
  - Оператор switch предполагает строгое равенство ===.

Условный оператор IF

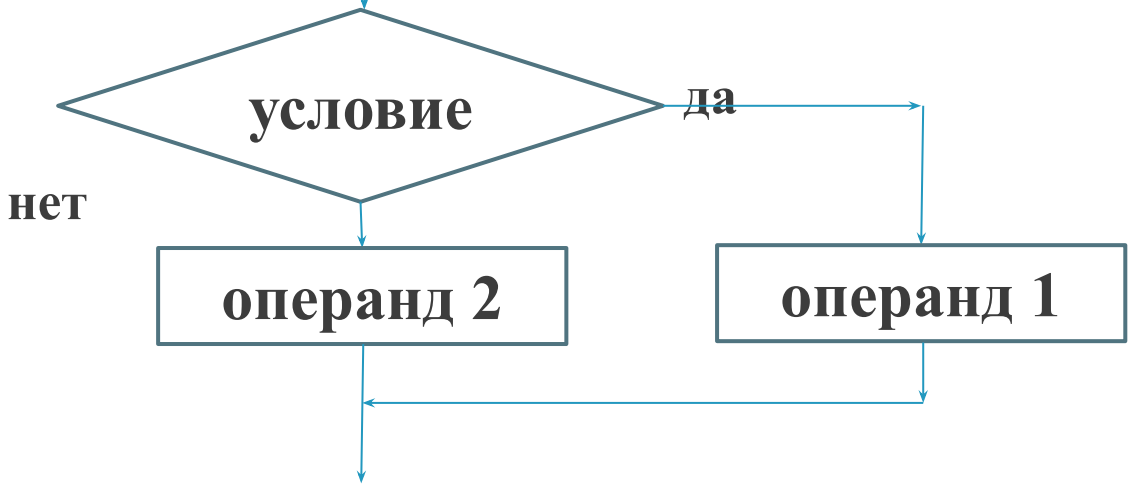
- С одной веткой

if (условие) {…}



- С двумя ветвями

if (условие) {…} else {…}



Условный оператор «?»

```
let result = условие ? значение1 : значение2;
```

Пример оператора «?»

```
let boolean_var = num > 72 ? true : false;
```

Пример Switch

```
Var a = prompt('Введите аргумент…');

Switch (a) {
   case '1': alert('Символ 1'); break;
   case '2': alert('Символ 2'); break;
   case 1   : alert('Число 1');    break;
   default  : alert('Что-то другое'); break;
}
```

Операторы цикла

- for (i = i1; i = i2; i++) {…}

```
<script>
    var P=1;
    for (i=1; i<=7; i++) {
      P=P*i;
    }
    alert((i-1)+"!="+P);
  </script>
```

Цикл while

- while(условие входа в цикл) {…}

```
P=1;
i=1;
while (i<8) {
  P=P*i;
}
alert((i-1)+"!="+P);
```

do - while

- do
  {…}
  while (условие нахождения в цикле)

```
P=1;
i=1;
do {
   P=P*i;
   i=i+1;
}
while (i<9);
alert((i-1)+"!="+P);
```

Досрочный выход из цикла

break

```
var sum = 0;
while (true) {
   var value = prompt("Введите число", "");
   if (value==7) break; // если введено 7
   sum += value;
}
alert( 'Сумма: ' + sum )
```

Досрочное завершение текущей итерации

continue

```
for (var i = 0; i < 16; i++) {
   if (i % 2 != 0) continue; // если нечетное число
   alert(i);
}
```

Функции

Функция – поименованный автономный блок программы, который может быть вызван к исполнению из любого места программы и осуществить возврат по исполнении в место вызова.

Функции:

- Пользователя
- Стандартные

Функции пользователя

```
function имя(параметры) {
  ...тело...
   return …возвращаемой значение
}
```

Функция как процедура

```
…
function my_proc() {
    alert(‘Всем привет!!!’);
}
…
my_proc();
…
```

Передача глобальных параметров

```
let my_str = ‘Всем привет!!!’;
…
function my_proc() {
    alert(my_str);
}
...
my_proc();
…
```

Передача данных через окно параметров

```
let my_str_glob = ‘Всем привет!!!’;
…
function my_proc(str) {
    alert(str);
}
...
my_proc(my_str_glob);
…
```

Функция как функция

Возвращение результата через имя функции

```
let d;
…
function my_proc(a,b) {
    let c;
    c = a + b;
    return c;
}
...
d = my_proc(5,7);
alert(d);
```

Объекты

```
let user = new Object(); // "конструктор объекта"


let user = { // "литерал объекта"
 ключ1: значение1,
 ключ2: значение2,
 ...
 ключN: значениеN
};
```

Пример объекта

```
let user = new Object();
let user = {     // объект
    name: "Вовочка",
    age: 21
};
...

alert(user.name+', '+user.age);
```

Функции объекта Math

- sin, cos, tan (угол в радианах)
- acos, asin, atan, atan2
- abs, floor, ceil, log, pow, max, min, round,
- random, sqrt

- a = Math.sin(1.32);
- With(Math) {

```
    a = sin(1.32);
    b = cos(1.21);
}
```