

## ЧТО ТАКОЕ CSS

Основным понятием CSS является стиль – т. е. набор правил оформления и форматирования, который может быть применен к различным элементам страницы. В стандартном HTML для присвоения какому-либо элементу определенных свойств (таких, как цвет, размер, положение на странице и т. п.) приходилось каждый раз описывать эти свойства, даже если на одной страничке должны располагаться 10 или 110 таких элементов, ничуть не отличающихся один от другого. Вы должны были десять или сто десять раз вставить один и тот же кусок HTML-кода в страничку, увеличивая размер файла и время загрузки на компьютер просматривающего ее пользователя.

CSS действует более удобным и экономичным способом. Для присвоения какому-либо элементу определенных характеристик вы должны один раз описать этот элемент и определить это описание как стиль, а в дальнейшем просто указывать, что элемент, который вы хотите оформить соответствующим образом, должен принять свойства стиля, описанного вами.

Более того, вы можете сохранить описание стиля не в тексте вашей странички, а в отдельном файле – это позволит использовать описание стиля на любом количестве Web-страниц, а также изменить оформление любого количества страниц, исправив лишь описание стиля в одном (отдельном) файле.

Кроме того, CSS позволяет работать со шрифтовым оформлением страниц на гораздо более высоком уровне, чем стандартный HTML, избегая излишнего утяжеления страниц графикой.

## ПРАКТИЧЕСКОЕ ОСВОЕНИЕ CSS

Существует целых четыре способа связывания документа и таблицы стилей:

1. Связывание — позволяет использовать одну таблицу стилей для форматирования многих страниц HTML
2. Внедрение — позволяет задавать все правила таблицы стилей непосредственно в самом документе
3. Встраивание в теги документа — позволяет изменять форматирование конкретных элементов страницы
4. Импортирование — позволяет встраивать в документ таблицу стилей, расположенную на сервере

Остановимся на каждом из этих способов более подробно.

### СВЯЗЫВАНИЕ

Как вам уже известно, информация о стилях может располагаться либо в отдельном файле, либо непосредственно в коде Web-странички. Расположение описания стилей в отдельном файле имеет смысл в случае, если вы планируете применять эти стили к большему, чем одна, количеству страниц. Для этого нужно создать обычный текстовый файл, описать с помощью языка CSS необходимые стили, разместить этот файл на Web-сервере, а в коде Web-страниц, которые будут использовать стили из этого файла, нужно будет сделать ссылку на него. Делается это с помощью тега LINK, располагающегося внутри тега HEAD ваших страниц:

```
<LINK REL=STYLESHEET TYPE="text/css" HREF="URL" >
```

Первые два параметра этого тега являются зарезервированными именами, требующимися для того, чтобы сообщить браузеру, что на этой страничке будет использоваться CSS. Третий параметр – HREF= «URL» – указывает на файл, который содержит описания стилей. Этот параметр должен содержать либо относительный путь к файлу – в случае, если он находится на том же сервере, что и документ, из которого к нему обращаются – или полный URL («http://...») в случае, если файл стилей находится на другом сервере.

### ВНЕДРЕНИЕ

Второй вариант, при котором описание стилей располагается в коде Web-странички, внутри тега HEAD, в теге <STYLE type="text/css">... </STYLE>. В этом случае вы можете использовать эти стили для элементов, располагающихся в пределах странички. Параметр type="text/css" является обязательным и служит для указания браузеру использовать CSS.

### ВСТРАИВАНИЕ В ТЕГИ ДОКУМЕНТА

Третий вариант, когда описание стиля располагается непосредственно внутри тега элемента, который вы описываете. Это делается с помощью параметра STYLE, используемого при применении CSS с большинством стандартных тегов HTML. Этот метод нежелателен, он приводит к потере одного из основных преимуществ CSS – возможности отделения информации от описания оформления информации. Впрочем, если необходимо описать лишь один элемент, этот вариант расположения описания стилей также вполне применим.

### ИМПОРТИРОВАНИЕ

В теге <STYLE> можно *импортировать* внешнюю таблицу стилей с помощью свойства @import таблицы стилей:

```
@import: url(mystyles.css);
```

Его следует задавать в начале стилевого блока или связываемой таблицы стилей перед заданием остальных

правил. Значение свойства @import является URL файла таблицы стилей.

## ГРУППИРОВАНИЕ

Правила каскадных таблиц состоят из селектора и определения. Для уменьшения размеров таблиц стилей можно *группировать* разные селекторы в виде списка элементов страницы HTML, разделённых запятыми, если для них задаётся одно правило. Например, следующие правила

```
H1 {font-family: Verdana}
```

```
H2 {font-family: Verdana}
```

можно сгруппировать и задать в виде одного правила со списком селекторов

```
H1, H2 {font-family: Verdana}
```

Аналогично группируются определения, только в списке они разделяются точками с запятой. Следующие правила форматирования заголовка первого уровня

```
H2 {font-weight: bold}
```

```
H2 {font-size: 14pt}
```

```
H2 {font-family: Verdana}
```

можно сгруппировать в виде одного правила, сгруппировав определения:

```
H2 (font-weight: bold; font-size: 14pt; font-family: Verdana;)
```

Некоторые свойства имеют собственный синтаксис группирования, связанный с заданием значений нескольких свойств в одном. Например, предыдущий пример при использовании свойства font запишется так:

```
H2 {font: bold 14pt Verdana}
```

При задании таблицы стилей можно свободно комбинировать все три правила группирования для уменьшения её размеров.

## НАСЛЕДОВАНИЕ

В HTML некоторые элементы могут содержать другие. Как будет отображаться элемент, расположенный внутри другого элемента страницы, если для последнего задано правило форматирования, а для вложенного элемента нет? Например, пусть цвет шрифта абзаца определён как синий (P {color: blue}). Как будет отображаться выделенный элемент текста, задаваемый тэгом <em>, если для него не определено правило форматирования? В подобных случаях вложенный элемент *наследует* правила форматирования элемента-родителя. В нашем примере выделенный элемент будет также отображаться синим цветом. Другие свойства ведут себя аналогично свойству color, например font-family, font-size.

Некоторые свойства не наследуются вложенными элементами от своих родителей, например свойство background, но по умолчанию вложенные элементы будут отображаться с фоном родительского элемента.

Наследование полезно при задании значений свойств, применяемых к документу по умолчанию. Для этого достаточно задать все свойства для элемента, порождающего все остальные элементы страницы HTML. Таким элементом является тело документа, определяемое тегом BODY:

```
BODY {color: black;
font-family: "Times New Roman";
background: url(picture.gif) white;
}
```

Приведённые правила задают форматирование документа по умолчанию: чёрным шрифтом гарнитуры Times New Roman с фоном, задаваемым графическим файлом picture.gif, или на белом фоне, если файл недоступен.

## СЕЛЕКТОРЫ

Правила каскадных таблиц стилей, в которых в качестве селектора используются теги HTML, влияют на отображение всех элементов заданного типа в документе. Следующее правило отображает без подчёркивания все ссылки в документе.

```
< STYLE TYPE="text/css">
<!--
  A {text-decoration:none; }
-->
</--STYLE>
```

## КЛАССЫ

А что делать, если нужно некоторые ссылки отобразить по-другому? CSS реализует возможность присваивать стили не всем одинаковым элементам страницы, а избирательно – для этого используется параметр CLASS = "имя

класса" или идентификатор ID=«имя элемента», присваивающиеся любому элементу страницы. Рассмотрим эти возможности подробнее.

Класс позволяет задать разные правила форматирования для одного элемента определённого типа или всех элементов документа. Имя класса указывается в селекторе правила после имени тега и отделяется от него точкой. Можно определить несколько правил форматирования для одного элемента и с помощью параметра CLASS соответствующего тега применять разные правила форматирования. Например, можно определить два класса для отображения заголовка первого уровня:

```
<STYLE TYPE="text/css">
<!--
H1.red {color: red; }
H1.blue {color:red; background-color: blue}
-->
</STYLE>
```

В тексте документа ссылка на соответствующий класс задаётся в параметре CLASS:

```
<H1 CLASS="red">Красный шрифт</H1>
<H1 CLASS="blue">Красный шрифт на синем фоне</H1>
```

В приведённом примере классы задавались для разного отображения элементов одного типа. Если класс должен применяться ко всем элементам документа, то в селекторе задаётся имя класса с лидирующей точкой без указания конкретного элемента:

```
<STYLE TYPE="text/css">
<!--
.red {color: red; }
.blue {color:red; background-color: blue}
-->
</STYLE>
```

Теперь два класса red и blue можно применять к любым элементам документа:

```
<P CLASS="red">Красный шрифт</P>
<P CLASS="blue">Красный шрифт на синем фоне</P>
```

Первый абзац отразится красным шрифтом, а второй — красным шрифтом на синем фоне.

## ПСЕВДОКЛАССЫ

В CSS есть такое понятие как псевдокласс. В отличие от обычного класса, действие псевдокласса распространяется не на весь текст, к которому применен данный стиль, а лишь на его часть и возможно лишь в определенном состоянии. Чтобы было понятнее, давайте разберем эффект, при котором ссылки подчеркиваются лишь при наведении на них курсора. Эффект достаточно распространенный, и его можно наблюдать в том числе и на этом сайте. Вот фрагмент таблицы стилей, который позволяет достигать вышеописанного эффекта:

```
A:link { color: red } /* unvisited link */
A:visited { color: blue } /* visited links */
A:active { color: lime } /* active links */
```

Верхняя строчка — это переопределение стандартного тега <a>, которое запрещает подчеркивать ссылки, а вот нижняя — это определение стиля для псевдокласса *hover*, который описывает стиль ссылки в момент, когда курсор находится над ней.

Определение первой строки абзаца:

```
P:first-line { font-style: small-caps }
```

Определение буквы в начале абзаца:

```
P:first-letter { font-size: 200%; float: left }
```

Заметьте, что и в том, и в другом случае действие стиля распространяется либо на определенное состояние (пользователь собирается щелкнуть по ссылке), либо на фрагмент текста (изменяется только первая буква абзаца). В этом и заключается смысл псевдоклассов.

## ИДЕНТИФИКАТОРЫ

Присвоение стилей с помощью идентификаторов применяется в случае, если данному идентификатору соответствует только один элемент на странице. Если элементов, которым необходимо присвоить такой стиль, несколько — это уже класс.

Параметр ID, как и параметр CLASS, не влияет на отображение браузером элемента HTML, но задаёт уникальное имя элемента, которое используется для ссылок на него в сценариях и таблицах стилей. Параметр ID можно применять к любому элементу документа.

Правила таблиц стилей регламентируют использование уникального идентификационного имени элемента в качестве селектора, предворяя его символом #: <STYLE TYPE="text/css">

```
<!--
#myID {letter-spacing: 1em; }
H1#form3 {color:red; background-color: blue}
-->
</STYLE>
<BODY>
<P ID=myID> Разрежённые слова в абзаце</P>
<H1 ID=form2>Чёрный заголовок</P>
```

В этом примере абзац идентифицирован именем myID в параметре ID, поэтому к нему применимо правило с селектором #myID. Второе правило в таблице стилей должно применяться к заголовку первого уровня с идентификатором form3. Такого элемента в нашем фрагменте нет, и поэтому заголовок form2 отображается с применением правила по умолчанию.

## ПРОСТЕЙШИЙ ПРИМЕР

Давайте рассмотрим механизм, с помощью которого стили присваиваются элементам Web-страниц. Самый простой случай присвоения какому-либо элементу определенного стиля выглядит так:

```
НАЗВАНИЕ_ЭЛЕМЕНТА {свойство: значение;},
```

Где НАЗВАНИЕ\_ЭЛЕМЕНТА – имя HTML-тега (H1, P, TD, A и т. д.), а параметры в фигурных скобках – список свойств элемента и присвоенных им значений.

Пример:

```
H1 {font-size: 30pt; color: blue;}
```

В этом примере всем заголовкам на странице, оформленным тегом H1, присваивается размер шрифта 30 пунктов и синий цвет.

## СВОЙСТВА ЭЛЕМЕНТОВ, УПРАВЛЯЕМЫХ С ПОМОЩЬЮ CSS

В настоящее время язык CSS насчитывает довольно большое количество свойств элементов HTML, которыми он может управлять.

- Свойства шрифта
- Цвет элемента и цвет фона
- Свойства текста
- Размеры элемента
- Свойства таблицы
- Единицы измерения

Итак, перейдем к изучению элементов CSS. Описание свойств элементов в CSS состоит из названия свойства с последующим присвоением ему определенного значения. Название свойства и его значение разделены двоеточием.

Указывая абсолютные, а не относительные размеры шрифтов, вы лишаете людей, просматривающих ваши странички, возможности увеличивать или уменьшать размер шрифтов с помощью специальной кнопки в браузере в соответствии с разрешением их дисплея и зрением. Шрифты будут отображаться только такого размера, который вы указали при написании странички.

Поэтому, если использование абсолютных размеров шрифтов не обусловлено художественным замыслом или коварным умыслом, рекомендую вам использовать для этих целей указание размеров в процентах. В этом случае размер шрифта будет меньше (больше) на указанное вами количество процентов, чем при оформлении его с помощью стандартного HTML-тега.

Есть еще одна небольшая, но очень полезная хитрость – это способ скрыть от устаревших браузеров описания стилей, располагающихся в теге <STYLE>, внутри раздела<HEAD>. Поскольку браузер был написан несколько лет назад, когда никакого CSS еще и в планах не было, он просто не поймет, что это такое написано внутри <STYLE>...</STYLE>, и выдаст все описания стилей на страничку, как обычный текст. Для того чтобы предотвратить это, необходимо заключить описания стилей в тег комментариев. Делается это очень просто.

```
<HEAD>
<STYLE type="text/css">
<!--
описания стилей
-- >
</STYLE>
```

```
</HEAD>
```

где

```
<!-- - тег, открывающий комментарий, а
--> - закрывающий.
```

Устаревшие браузеры посчитают все заключенное между тегами комментариев информацией не отображенной, а новые и сообразительные браузеры определяют, что это описание стилей, и задействуют их.

Еще один из интересных вариантов применения CSS скрывается за, казалось бы, простой возможностью: вы можете указывать значения отступов вокруг объектов, как отрицательные величины! Это позволяет накладывать один слой текста на другой и получать весьма интересные и привлекательные результаты.

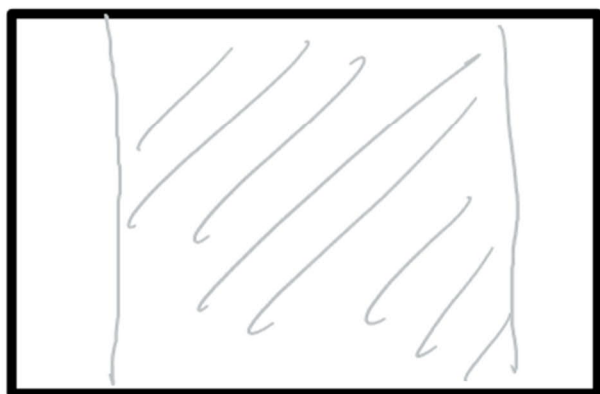
Внутри описания стиля для удобства форматирования вы можете использовать любое количество пробелов и переносов строк – при чтении стиля браузер просто отбросит все лишние пробелы.

## Основные параметры шрифта

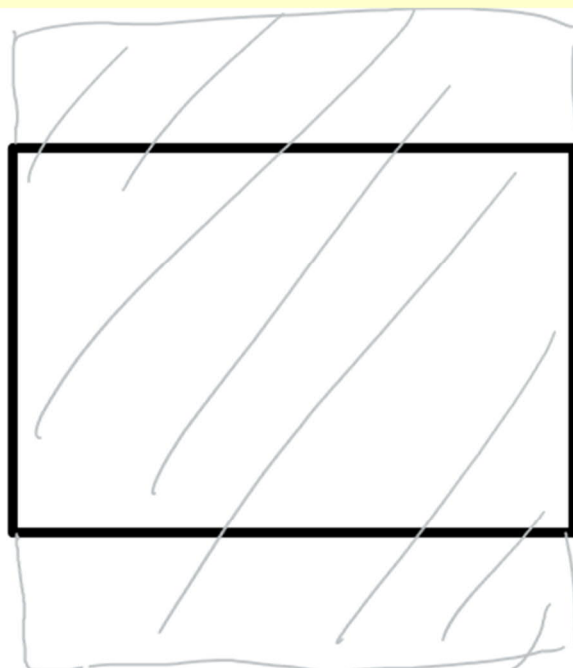
```
font-weight: [bold|normal|number] – жирность шрифта
font-style: [normal|italic|oblique] – наклон шрифта
font-size: number – размер шрифта
font-family: name – гарнитура шрифта
font: [font-style font-weight font-size font-family]
font: normal normal 14px serif;
color: number – цвет шрифта
```

## Параметры фона

```
background-color: number – цвет подложки
background-attachment: (fixed | scroll) – скроллинг
background-image: url(URL) – фоновое изображение
background-repeat: (repeat | no-repeat | repeat-x | repeat-y) – повторение фона
background-position: (left | right | center) (top | bottom | middle) – расположение фона
background-size: число число | % % | auto auto;
background-size: cover – масштабируется без изменения пропорций так, чтобы закрыть фон всего элемента
background-size: contain – масштабируется без изменения пропорций так, чтобы полностью поместиться в элемент
```



contain



cover

## Основные параметры абзаца/блока

```
text-align: [left|right|center|justify] – выравнивание
text-indent: number – отступ красной строки
line-height: number – интерлиньяж
letter-spacing: number – трекинг
text-transform: capitalize | lowercase | uppercase | none | inherit
word-spacing: <размер> | normal | inherit
letter-spacing: значение | normal | inherit
text-decoration: blink | line-through | overline | underline | none
    text-decoration-line: underline;
    text-decoration-thickness: 3px;
    text-decoration-style: solid | double | dotted | dashed | wavy;
    text-decoration-color: red;
```

## Отступы

```
padding-left: number – отступ от текста слева
padding-right: number – отступ от текста справа
padding-top: number – отступ от текста сверху
padding-bottom: number – отступ от текста снизу
padding: [значение | проценты] {1, 4} | inherit

margin-left: number – отступ от границы слева
margin-right: number – отступ от границы справа
margin-top: number – отступ от границы сверху
margin-bottom: number – отступ от границы снизу
margin: [значение | проценты | auto] {1,4} | inherit
```

## Ссылки

```
a:link {color: blue} – теговый класс для ссылки
a:hover {color: red} – теговый класс для обработчика событий – наведение на ссылку
a: active {*} – теговый класс для обработчика событий – активная ссылка
```

## Списки

```
list-style-type: circle | disc | square | decimal;
list-style-position: inside | outside;
list-style-image: url('путь и имя файла картинки-маркера')
list-style: [type image position]
list-style: decimal url('marker.png') outside
```

<ul>

<li> elements

<li> elements

</ul>

<ol>

<li> elements

<li> elements

</ol>

## Рамка

```
border-width: [значение | thin | medium | thick] {1,4} | inherit
border-width: 3px 7px 7px 4px;
border-style: [none | hidden | dotted | dashed | solid | double | groove | ridge |
inset | outset] {1,4} | inherit;
border-color: [цвет | transparent] {1,4} | inherit;
border: [border-width || border-style || border-color];
border: 1px solid black;
border-radius: число;
border-radius: 40px 10px;
border-radius: 10px;
```

## Картинки

```
img {
  display: block;
  width: 350px;
  height: 350px;
  object-fit: cover;
  object-position: 80% top;
}
```

```
/* *****
MEDIA QUERIES
***** */

/* for 980px or less */
@media screen and (max-width: 980px) {
}

/* for 700px or less */
@media screen and (max-width: 700px) {
}

/* for 480px or less */
@media screen and (max-width: 480px) {
}
@media print {
}

@page {
  margin: 2cm;
}
```



## МОДЕЛЬ БОКСА

### РАЗМЕРЫ БОКСА

Каждый бокс имеет *область содержимого* (например, текст, изображение и т.п.) и необязательное окружение — области *заполнения*, *рамки* и *поля*; размер каждой области специфицируется свойствами, определёнными ниже. На диаграмме показано соотношение этих областей и терминология, используемая для ссылок на разные участки поля/margin, рамки/border и заполнения/padding:

Периметр каждой из четырёх областей (содержимого, заполнения, рамки и поля) называется "край", соответственно — каждый бокс имеет четыре края:

#### **content edge/край содержимого** или **inner edge/внутренний край**

Край содержимого окружает отображаемое содержимое.

#### **padding edge/край заполнения**

Окружает заполнение бокса. Если заполнение имеет ширину 0, край заполнения — это то же, что и край содержимого. Край заполнения бокса определяет края содержащего блока, установленного боксом.

#### **border edge/край рамки**

Окружает рамку бокса. Если рамка имеет ширину 0, то край рамки — это то же, что и край заполнения.

#### **margin edge/край поля** или **outer edge/внешний край**

Окружает поле бокса. Если поле имеет ширину 0, то край поля — тот же, что и край рамки.

Каждый край может быть разорван слева, справа, сверху и внизу.

Размеры области содержимого бокса — *ширина содержимого* и *высота содержимого* — зависят от нескольких факторов: имеет ли элемент, генерирующий бокс, установленные свойства `'width'` или `'height'`, содержит ли бокс текст или другие боксы, является ли бокс таблицей и т.д. Ширина и высота бокса обсуждаются в главе *некоторые детали модели визуального форматирования*.

*Ширина бокса* выводится как сумма левого и правого поля, рамки, заполнения и ширины содержимого. *Высота* выводится как сумма верхнего и нижнего поля, рамки, заполнения и высоты содержимого.

Стиль фона различных областей бокса определяется так:

- *Область содержимого*: свойство `'background'` генерирующего элемента.
- *Область заполнения*: свойство `'background'` генерирующего элемента.
- *Область рамки*: свойства рамки генерирующего элемента.
- *Область поля*: поля всегда прозрачны.

### ПРИМЕР ПОЛЕЙ, ЗАПОЛНЕНИЯ И РАМОК

Этот пример документа HTML показывает, как поля, рамки и заполнение взаимодействуют:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
  <HEAD>
    <TITLE>Пример полей, заполнения и рамок</TITLE>
    <STYLE type="text/css">
      UL {
        background: green;
        margin: 12px 12px 12px 12px;
        padding: 3px 3px 3px 3px;
                                          /* Рамки не установлены */
      }
      LI {
        color: black;                      /* цвет текста — чёрный */
        background: gray;                 /* Содержимое, заполнение будет серым */
        margin: 12px 12px 12px 12px;
        padding: 12px 0px 12px 12px;      /* Заметьте, что заполнение справа 0px */
        list-style: none                  /* перед элементом списка нет никаких глифов */
      }
                                          /* Рамки не установлены */
      LI.withborder {
        border-style: dashed;
        border-width: medium;             /* устанавливает ширину рамок всех сторон */
        border-color: black;
      }
    </STYLE>
```



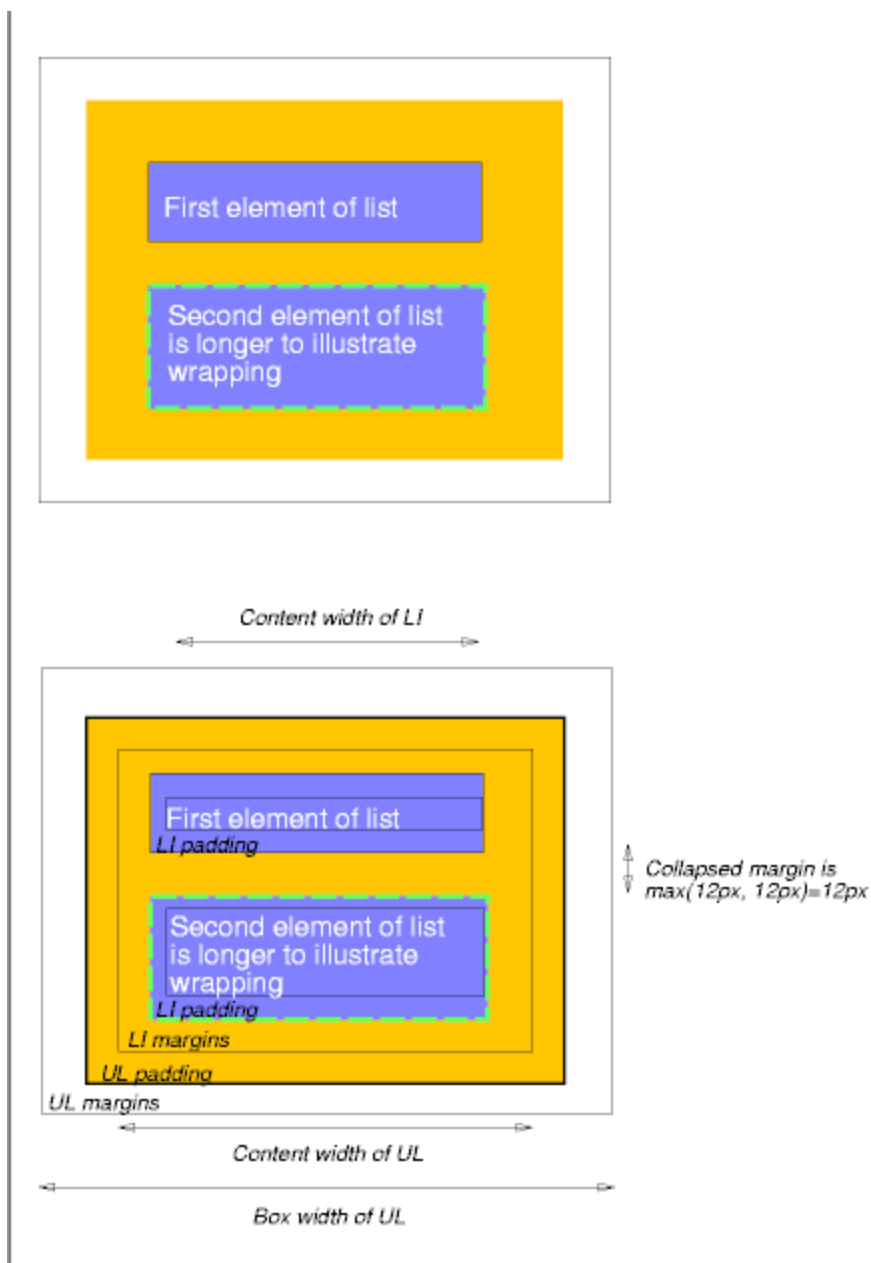
```

</HEAD>
<BODY>
  <UL>
    <LI>Первый элемент списка
    <LI class="withborder">Второй элемент списка длиннее,
                           чтобы показать перенос.
  </UL>
</BODY>
</HTML>

```

и даёт дерево документа с (помимо других соотношений) элементом UL, в котором есть два дочерних LI.

Первая диаграмма иллюстрирует, что этот пример даст в результате. Вторая иллюстрирует взаимоотношения между полями, рамками и заполнением элементов UL и LI.



Обратите внимание, что:

- Ширина содержимого каждого бокса LI вычисляется сверху вниз; содержащий блок для каждого бокса LI устанавливается элементом UL.
- Высота каждого бокса LI задаётся высотой содержимого плюс верхнее и нижнее заполнение, рамки и поля. Обратите внимание, что горизонтальные поля между боксами LI сжаты.
- Правое заполнение боксов LI было установлено шириной 0 (свойство ['padding'](#)). Результат виден на второй иллюстрации.
- Поля боксов LI прозрачны — поля всегда прозрачны — так что цвет фона (зелёный) заполнения и области содержимого UL просвечивает сквозь него.
- Второй элемент LI специфицирует пунктирную рамку (свойство ['border-style'](#)).

## СВОЙСТВА ПОЛЯ: 'MARGIN-TOP', 'MARGIN-RIGHT', 'MARGIN-BOTTOM', 'MARGIN-LEFT' И 'MARGIN'

Свойства поля специфицируют [область полей](#) бокса. Сокращённое свойство '[margin](#)' устанавливает поле для всех четырёх сторон, в то время как другие свойства устанавливают только соответствующие стороны.

Свойства, определённые в этом разделе, относятся к типу значений `<margin-width>`, который может устанавливаться в:

### [<length>](#)

Специфицирует фиксированную ширину.

### [<percentage>](#)

Процентное значение высчитывается относительно *ширины содержащего блока* сгенерированного бокса.

Это верно для '[margin-top](#)' и '[margin-bottom](#)' везде, кроме [контекста страницы](#), где процентные значения относятся к высоте бокса страницы.

Негативные значения свойств полей допускаются, но могут существовать ограничения, специфичные для конкретных реализаций.

### 'margin-top', 'margin-right', 'margin-bottom', 'margin-left'

Значение:	<a href="#">&lt;margin-width&gt;</a>   <a href="#">inherit</a>
Начальное:	0
Применяется:	ко всем элементам
Наследуется:	нет
Процентное:	относительно ширины содержащего блока
Носитель:	<a href="#">визуальный</a>

Это свойство устанавливает верхнее, нижнее, правое и левое поля бокса.

```
H1 { margin-top: 2em }
```

### 'margin'

Значение:	<a href="#">&lt;margin-width&gt;</a> {1,4}   <a href="#">inherit</a>
Начальное:	не определено для сокращённого свойства
Применяется:	ко всем элементам
Наследуется:	нет
Процентное:	относительно ширины содержащего блока
Носитель:	<a href="#">визуальный</a>

'[margin](#)' — это сокращённое свойство для установки значений '[margin-top](#)', '[margin-right](#)', '[margin-bottom](#)' и '[margin-left](#)' в одном месте в таблице стилей.

Если имеется одно значение, то оно применяется ко всем сторонам. Если дано два значения, верхнее и нижнее поля устанавливаются в первое, а правое и левое поля — во второе значение. Если дано три значения, верхнее поле устанавливается в первое, левое и правое — во второе, а нижнее поле — в третье значение. Если задано четыре значения, они применяются к верхнему, правому, нижнему и левому полям соответственно.

```
BODY { margin: 2em } /* все поля установлены в 2em */
BODY { margin: 1em 2em } /* верхнее и нижнее = 1em, правое и левое = 2em */
BODY { margin: 1em 2em 3em } /* верхнее=1em, правое=2em, нижнее=3em, левое=2em */
Последнее правило эквивалентно следующему:
BODY {
  margin-top: 1em;
  margin-right: 2em;
  margin-bottom: 3em;
  margin-left: 2em; /* копируется из противоположной стороны (из правой)
*/
}
```

## СЖАТИЕ ПОЛЕЙ

В этой спецификации выражение *сжатие полей* означает, что смежные поля (не разделённые заполнением и рамками) двух или более боксов (которые могут быть последовательными или вложенными) образуют единое поле.

В CSS2 вертикальные поля никогда не сжимаются.

Горизонтальные поля могут сжиматься между определёнными боксами:

- Два или более смежных горизонтальных поля боксов [блока](#) при [нормальном расположении](#) сжимаются. Результирующая ширина поля — это максимальная ширина из смежных полей. В случае негативных значений полей, абсолютный максимум негативных смежных полей отсчитывается от максимума позитивных смежных полей. Если нет позитивных полей, абсолютный максимум негативных смежных полей отсчитывается от нуля.
- Вертикальные поля между обтекаемым боксом и любым другим боксом не сжимаются.
- Поля абсолютно и относительно позиционированных боксов не сжимаются.

## СВОЙСТВА ЗАПОЛНЕНИЯ: 'PADDING-TOP', 'PADDING-RIGHT', 'PADDING-BOTTOM', 'PADDING-LEFT' И 'PADDING'

Свойства обтекания специфицируют ширину [области заполнения](#) бокса. Сокращённое свойство ['padding'](#) устанавливает заполнение для всех четырёх сторон, а другие свойства заполнения устанавливают соответствующие стороны.

Свойства, определённые в этом разделе, относятся к типу значений `<padding-width>`, который может устанавливаться в:

### [<length>](#)

Специфицирует фиксированную ширину.

### [<percentage>](#)

Процентное, высчитывается относительно *ширины* содержащего блока, генерирующего бокс, в том числе и для ['padding-top'](#) и ['padding-bottom'](#).

В отличие от свойств полей, значения заполнения не могут быть негативными. Как и в свойствах полей, процентные значения свойств заполнения относятся к ширине бокса, сгенерированного содержащим блоком.

### 'padding-top', 'padding-right', 'padding-bottom', 'padding-left'

Значение:	<a href="#">&lt;padding-width&gt;</a>   <a href="#">inherit</a>
Начальное:	0
Применяется:	ко всем элементам
Наследуется:	нет
Процентное:	относительно ширины содержащего блока
Носитель:	визуальный

Эти свойства устанавливают верхнее, правое, нижнее и левое заполнение для бокса.

```
BLOCKQUOTE { padding-top: 0.3em }
```

### 'padding'

Значение:	<a href="#">&lt;padding-width&gt;</a> {1,4}   <a href="#">inherit</a>
Начальное:	не определено для сокращённого свойства
Применяется:	ко всем элементам
Наследуется:	нет
Процентное:	относительно ширины содержащего блока
Носитель:	визуальный

Свойство ['padding'](#) — это сокращённое свойство для установки ['padding-top'](#), ['padding-right'](#), ['padding-bottom'](#) и ['padding-left'](#) в одном месте в таблице стилей.

Если имеется одно значение, то оно применяется ко всем сторонам. Если дано два значения, верхнее и нижнее заполнение устанавливаются в первое, а правое и левое заполнения — во второе значение. Если дано три значения, верхнее заполнение устанавливается в первое, левое и правое — во второе, а нижнее заполнение — в третье значение. Если задано четыре значения, они применяются к верхнему, правому, нижнему и левому заполнениям соответственно.

Цвет поверхности области заполнения специфицируется через свойство ['background'](#):

```
H1 {
  background: white;
  padding: 1em 2em;
}
```

Вышеприведённый пример специфицирует вертикальное заполнение '1em' (['padding-top'](#) и ['padding-bottom'](#)) и

горизонтальное заполнение '2em' ('padding-right' и 'padding-left'). Единицы измерения 'em' относительно к размеру шрифта элемента: '1em' эквивалентно размеру используемого шрифта.

## СВОЙСТВА РАМКИ

Свойства рамки специфицируют ширину, цвет и стиль [области рамки](#) бокса. Эти свойства применимы ко всем элементам.

*Примечание.* Особенно в HTML, ПА могут отображать рамки определённых элементов (например, кнопок, меню и т.п.) иначе, чем у "обычных" элементов.

**Ширина рамки:** ['border-top-width'](#), ['border-right-width'](#), ['border-bottom-width'](#), ['border-left-width'](#) и ['border-width'](#)

Свойства ширины рамки специфицируют ширину [области рамки](#). Свойства, определённые в этом разделе, относятся к типу значений **<border-width>**, который может устанавливаться в:

### thin

Тонкая рамка.

### medium

Средняя рамка.

### thick

Толстая рамка.

### [<length>](#)

Толщина рамки имеет точное значение. Это значение не может быть негативным.

Интерпретация первых трёх значений зависит от ПА. Следующие соотношения, однако, обязаны выдерживаться:

```
'thin' <= 'medium' <= 'thick'.
```

К тому же эти значения ширины обязаны быть константными в пределах документа.

**'border-top-width', 'border-right-width', 'border-bottom-width', 'border-left-width'**

Значение: [<border-width>](#) | [inherit](#)

Начальное: medium

Применяется: ко всем элементам

Наследуется: нет

Процентное: N/A

Носитель: [визуальный](#)

Эти свойства устанавливают верхнюю, правую, нижнюю и левую линии рамки для бокса.

**'border-width'**

Значение: [<border-width>](#){1,4} | [inherit](#)

Начальное: см. индивидуальные свойства

Применяется: ко всем элементам

Наследуется: нет

Процентное: N/A

Носитель: [визуальный](#)

Это сокращённое свойство для установки ['border-top-width'](#), ['border-right-width'](#), ['border-bottom-width'](#) и ['border-left-width'](#) в одном месте в таблице стилей.

Если имеется одно значение, то оно применяется ко всем сторонам. Если дано два значения, верхняя и нижняя линии рамки устанавливаются в первое, а правая и левая линии рамки — во второе значение. Если дано три значения, верхняя линия рамки устанавливается в первое, левая и правая — во второе, а нижняя линия рамки — в третье значение. Если задано четыре значения, они применяются к верхней, правой, нижней и левой линиям рамки соответственно.

Комментарии в данном примере поясняют результаты установки ширины верхней, правой, нижней и левой линий рамки:

```
H1 { border-width: thin }           /* thin thin thin thin */
H1 { border-width: thin thick }    /* thin thick thin thick */
H1 { border-width: thin thick medium } /* thin thick medium thick */
```

## ЦВЕТ РАМКИ: 'BORDER-TOP-COLOR', 'BORDER-RIGHT-COLOR', 'BORDER-BOTTOM-COLOR', 'BORDER-LEFT-COLOR' И 'BORDER-COLOR'

Эти свойства специфицируют цвет рамки бокса.

**'border-top-color', 'border-right-color', 'border-bottom-color', 'border-left-color'**

Значение:	<a href="#">&lt;color&gt;</a>   <a href="#">inherit</a>
Начальное:	значение свойства 'color'
Применяется:	ко всем элементам
Наследуется:	нет
Процентное:	N/A
Носитель:	<a href="#">визуальный</a>

**'border-color'**

Значение:	<a href="#">&lt;color&gt;</a> {1,4}   transparent   <a href="#">inherit</a>
Начальное:	см. индивидуальные свойства
Применяется:	ко всем элементам
Наследуется:	нет
Процентное:	N/A
Носитель:	<a href="#">визуальный</a>

Свойство ['border-color'](#) устанавливает цвет рамки с четырёх сторон. Значения имеют следующий смысл:

[<color>](#)

Специфицирует значение цвета.

**transparent**

Рамка прозрачна (хотя и может иметь ширину).

Свойство ['border-color'](#) может иметь от одного до четырёх значений, и эти значения устанавливаются для четырёх сторон так же, как и в 'border-width'.

Если цвет рамки элемента не специфицирован в свойстве рамки, ПА обязаны использовать значение свойства ['color'](#) элемента как [вычисленное значение](#) цвета рамки.

В этом примере рамка будет сплошной и чёрной:

```
P {
  color: black;
  background: white;
  border: solid;
}
```

## СТИЛЬ РАМКИ: 'BORDER-TOP-STYLE', 'BORDER-RIGHT-STYLE', 'BORDER-BOTTOM-STYLE', 'BORDER-LEFT-STYLE' И 'BORDER-STYLE'

Свойства стиля рамки специфицируют стиль линии рамки бокса (solid, double, dashed и т.д.). Свойства, определённые в этом разделе, относятся к типу значений **<border-style>**, который может устанавливаться в:

**none**

Нет рамки. Форсирует вычисленное значение 'border-width' в '0'.

**hidden**

То же, что и 'none', за исключением [разрешения конфликтов рамок](#) для [элементов таблицы](#).

**dotted**

Рамка из точек.

**dashed**

Рамка из пунктирных линий.

**solid**

Рамка из сплошной линии.

**double**

Двойная сплошная линия. Сумма двух линий и пространства между ними равна значению 'border-width'.

**groove**

Рамка выглядит как вырезанная в канве.

**ridge**

Противоположно 'groove': рамка выглядит как выступающая над канвой.

**inset**

Весь бокс выглядит вдавленным в канву.

**outset**

Противоположно 'inset': выпуклый бокс.

Все рамки прорисовываются на поверхности фона бокса. Цвет рамок со значениями 'groove', 'ridge', 'inset' и 'outset' зависит от свойства ['color'](#) элемента.

Соответствующие пользовательские агенты (ПА) HTML могут интерпретировать 'dotted', 'dashed', 'double', 'groove', 'ridge', 'inset' и 'outset' как 'solid'.

**'border-top-style', 'border-right-style', 'border-bottom-style', 'border-left-style'**

Значение: [<border-style>](#) | [inherit](#)

Начальное: none

Применяется: ко всем элементам

Наследуется: нет

Процентное: N/A

Носитель: [визуальный](#)

**'border-style'**

Значение: [<border-style>](#){ 1,4 } | [inherit](#)

Начальное: см. индивидуальные свойства

Применяется: ко всем элементам

Наследуется: нет

Процентное: N/A

Носитель: [визуальный](#)

Свойство ['border-style'](#) устанавливает стиль для четырёх сторон рамки. Может иметь от одного до четырёх значений, и значения устанавливаются для разных сторон, как для 'border-width' выше.

```
#xy34 { border-style: solid dotted }
```

В этом примере горизонтальные линии рамки будут 'solid', а вертикальные — 'dotted'.

Поскольку начальное значение стиля рамки — 'none', рамка будет видна только после установки значения стиля.

**СОКРАЩЁННЫЕ СВОЙСТВА РАМОК: ['BORDER-TOP'](#), ['BORDER-BOTTOM'](#), ['BORDER-RIGHT'](#), ['BORDER-LEFT'](#) И ['BORDER'](#)****'border-top', 'border-right', 'border-bottom', 'border-left'**

Значение: [ [<border-top-width>](#) || [<border-style>](#) || [<color>](#) ] | [inherit](#)

Начальное: см. индивидуальные свойства

Применяется: ко всем элементам

Наследуется: нет

Процентное: N/A

Носитель: [визуальный](#)

Это сокращённое свойство для установки ширины, стиля и цвета верхней, правой, нижней и левой сторон рамки.

```
H1 { border-bottom: thick solid red }
```

Это правило устанавливает ширину, стиль и цвет рамки **после** элемента H1. Опушенные значения установлены в свои [начальные значения](#). Поскольку нижеследующее правило не специфицирует цвет рамки, рамка будет иметь цвет, определённый свойством ['color'](#):

```
H1 { border-bottom: thick solid }
```

**'border'**

Значение: [ [<border-width>](#) || [<border-style>](#) || [<color>](#) ] | [inherit](#)

Начальное: см. индивидуальные свойства

Применяется: ко всем элементам

Наследуется: нет

Процентное: N/A

Носитель: [визуальный](#)

Свойство `'border'` — это сокращённое свойство для установки одинаковых значений ширины, цвета и стиля для всех четырёх сторон рамки бокса. В отличие от сокращённых свойств `'margin'` и `'padding'`, свойство `'border'` не может устанавливать разные значения для четырёх сторон. Для этого придётся использовать одно или более других свойств.

Первое правило эквивалентно установке четырёх последующих значений:

```
P { border: solid red }
P {
  border-top: solid red;
  border-right: solid red;
  border-bottom: solid red;
  border-left: solid red
}
```

Поскольку, в некоторой степени, свойства могут перекрываться, порядок, в котором правила специфицированы, имеет важное значение.

```
BLOCKQUOTE {
  border-color: red;
  border-left: double;
  color: black
}
```

В приведённом примере цвет левой линии рамки — чёрный, а других линий — красный. Это потому, что в `'border-left'` установлены ширина, стиль и цвет. Поскольку значение цвета в свойстве `'border-left'` не установлено, оно будет взято из свойства `'color'`. Фактически свойство `'color'`, установленное после свойства `'border-left'`, не имеет к этому никакого отношения.

## БЛОЧНОЕ ФОРМАТИРОВАНИЕ

Структура блоков определяется:

- [размерами боксов](#) и [типом](#).
- [схемой позиционирования](#) (нормальное позиционирование, всплывание и абсолютное).
- взаимоотношениями между элементами в [дереве документа](#).
- внешней информацией (например, размером порта просмотра, [внутренними](#) размерами изображений и т.д.).

Ширина начального содержащего блока может быть специфицирована свойством `'width'`. Если это свойство имеет значение `'auto'`, использует текущую ширину [экрана](#).

Высота начального содержащего блока может быть специфицирована свойством `'height'` корневого элемента. Если это свойство имеет значение `'auto'`, высота содержащего блока увеличивается, чтобы приспособиться к содержимому документа.

## СВОЙСТВО 'WIDTH'

Значение: [<length>](#) | [<percentage>](#) | `auto` | [inherit](#)

Начальное: `auto`

Применяется: ко всем элементам, кроме незамещаемых инлайн-элементов, рядов таблиц и групп рядов

Наследуется: нет

Процентное: относительно ширины содержащего блока

Носитель: [визуальный](#)

Значения имеют следующий смысл:

[<length>](#) — фиксированная ширина в рх.

[<percentage>](#) — ширина в процентах. Проценты вычисляются относительно ширины [содержащего блока](#) генерируемого бокса.

`auto` — Ширина зависит от значений других блоков, с учетом отступов.



['width'](#), ['margin-left'](#), ['margin-right'](#), ['left'](#) и ['right'](#)

Отрицательные значения для ['width'](#) не допускаются.

Это правило фиксирует ширину содержимого параграфа в 100 пикселей:

```
P { width: 100px }
```

## СВОЙСТВО 'MIN-WIDTH' И 'MAX-WIDTH'

### 'min-width'

Значение: [<length>](#) | [<percentage>](#) | [inherit](#)

### 'max-width'

Значение: [<length>](#) | [<percentage>](#) | none | [inherit](#)

Эти два свойства позволяют авторам ограничить ширину бокса определёнными рамками.

Значения имеют следующий смысл:

#### [<length>](#)

Специфицирует фиксированную минимальную и максимальную вычисленную ширину.

#### [<percentage>](#)

Специфицирует проценты для определения вычисленного значения. Проценты высчитываются относительно ширины [содержащего блока](#) генерируемого бокса.

#### none

(Только для ['max-width'](#)) Нет ограничений на ширину бокса.

Следующий алгоритм описывает, как эти два свойства воздействуют на [вычисленное значение](#) свойства ['width'](#):

## СВОЙСТВО 'HEIGHT'

### 'height'

Значение: [<length>](#) | [<percentage>](#) | auto | [inherit](#)

Свойство определяет [высоту содержимого](#) боксов, генерируемых элементами уровня блока и [замещаемыми](#) элементами.

Это свойство не применяется к незамещаемым элементам [инлайн-уровня](#). Высота боксов незамещаемых инлайн-элементов задаётся значением (возможно, наследуемым) ['line-height'](#) элемента.

Значения имеют следующий смысл:

#### [<length>](#)

Специфицирует фиксированную высоту.

#### [<percentage>](#)

Специфицирует высоту в процентах. Проценты высчитываются относительно высоты [содержащего блока](#) генерируемого бокса. Если высота содержащего блока не специфицирована явно (т.е. зависит от высоты содержимого), значение интерпретируется как 'auto'.

#### auto

Высота зависит от значений других свойств. См. ниже.

Отрицательные значения ['height'](#) недопустимы.

Следующее правило фиксирует высоту параграфа в 100 пикселей:

```
P { height: 100px }
```

Параграф, требующий высоты более 100 пикселей, будет вызывать [переполнение](#) в соответствии со свойством ['overflow'](#).

## СВОЙСТВО 'DISPLAY'

### 'display'

Значение: inline | block | list-item | run-in | compact | marker | table | inline-table | table-row-group | table-header-group | table-footer-group | table-row | table-column-group | table-column | table-cell | table-caption | none | [inherit](#)

Начальное:  
inline

Применяется:  
все элементы

Наследуется:	нет
Процентное:	N/A
Носитель:	<a href="#">все</a>

Значения этого свойства имеют следующий смысл:

**block** — Элемент генерирует основной бокс блока.

**inline** — Элемент генерирует один или более инлайн-боксов.

**list-item** — Элемент (например, LI в HTML) генерирует основной бокс блока и инлайн-бокс элемента списка. О списках и примерах форматирования списков см. раздел [списки](#).

**none** — Элемент **не** генерирует боксы в [структуре форматирования](#) (т.е. элемент не влияет на вид документа). Элементы-потомки не генерируют никаких боксов; это поведение **не может** быть переопределено установкой у потомков свойства 'display'.

Обратите внимание, что отображение 'none' не создаёт невидимый бокс; боксы вообще не создаются. CSS содержит механизмы, делающие возможным генерацию элементом бокса в структуре форматирования, который влияет на структуру форматирования, но невидим.

**run-in** и **compact** — Эти значения создают боксы блока и инлайн, в зависимости от контекста. Свойства, применяемые к боксам run-in и compact, базируются на окончательном статусе боксов (уровень инлайн или блока). Например, свойство 'white-space' применяется только тогда, когда бокс получает уровень блока.

**table**, **inline-table**, **table-row-group**, [table-column](#), **table-column-group**, **table-header-group**, **table-footer-group**, **table-row**, **table-cell** и **table-caption**

При установке этих значений, элемент ведёт себя как элемент таблицы.

Несколько примеров свойства 'display':

P	{ display: block }	
EM	{ display: inline }	
LI	{ display: list-item }	
IMG	{ display: none }	/* Не выводить изображения */

[Соответствующие](#) ПА HTML могут [игнорировать](#) свойство 'display'.

## СХЕМЫ ПОЗИЦИОНИРОВАНИЯ

В CSS2 бокс может находиться в разных слоях в соответствии со *схемами позиционирования*:

1. [Normal flow/Нормальное расположение](#). В CSS2 нормальное расположение включает [форматирование блока](#) для [боксов блока](#), [инлайн-форматирование](#) для [инлайн-боксов](#), [относительное позиционирование](#) боксов блока или инлайн и позиционирование [compact](#) и [run-in](#) боксов.
2. [Floats/Поплавки](#). В модели поплавок бокс сначала накладывается в соответствии с нормальным расположением, затем изымается из расположения и сдвигается влево или вправо, насколько возможно. Содержимое может обтекать по стороне "всплытия" поплавок.
3. [Абсолютное позиционирование](#). В модели абсолютного позиционирования бокс удаляется из нормального расположения полностью (это не действует на последующие родственные элементы) и получает позиционирование относительно содержащего блока.

**Примечание.** Схемы позиционирования CSS2 помогают авторам сделать документы более доступными, позволяя избегать трюков разметки (например, невидимых изображений), используемых для создания эффектов отображения.

## ВЫБОР СХЕМЫ ПОЗИЦИОНИРОВАНИЯ: СВОЙСТВО '[POSITION](#)'

Свойства '[position](#)' и '[float](#)' определяют, какой из алгоритмов позиционирования CSS2 используется для расчёта позиции бокса.

**'position'**

Значение:	static   relative   absolute   fixed   <a href="#">inherit</a>
Начальное:	static
Применяется:	ко всем элементам, но не к генерируемому содержимому
Наследуется:	нет
Процентное:	N/A

Носитель: [визуальный](#)

Значения имеют следующий смысл:

#### **static**

Бокс является нормальным боксом, расположенным в соответствии с [нормальным расположением](#). Свойства ['left'](#) и ['top'](#) не применяются.

#### **relative**

Позиция бокса высчитывается в соответствии с [нормальным расположением](#). Затем бокс смещается [относительно](#) нормального расположения. Если бокс В позиционирован относительно, расположение последующего бокса высчитывается так, как если бы бокс В не имел смещения.

#### **absolute**

Позиция бокса (и возможные размеры) определяется свойствами ['left'](#), ['right'](#), ['top'](#) и ['bottom'](#). Эти свойства устанавливают смещение относительно бокса [содержащего блока](#). Абсолютно позиционированные боксы изымаются из нормального обтекания. Это значит, что они не влияют на вывод последующих родственных элементов. Также, хотя [абсолютно позиционированные](#) боксы имеют поля, они не [соединяются](#) с другими полями.

#### **fixed**

Позиция бокса высчитывается в соответствии с моделью 'absolute', но, в дополнение к этому, бокс [фиксируется](#) в соответствии с некоторой ссылкой. В случае с [непрерывными носителями](#), бокс фиксируется относительно [порта просмотра](#) (и не перемещается при прокрутке). В случае со [страничными носителями](#), бокс фиксируется относительно страницы, даже если страница просматривается через [порт просмотра](#) (в случае просмотра перед печатью, например). Авторам может понадобиться специфицировать 'fixed' способом, не зависящим от носителя. Например, автор может захотеть, чтобы бокс оставался в верхней части [порта просмотра](#) экрана, но не вверху каждой печатаемой страницы. Две такие спецификации можно разделить, используя правило [@media](#), как здесь:

```
@media screen {
  H1#first { position: fixed }
}
@media print {
  H1#first { position: static }
}
```

### **ПОЗИЦИОНИРОВАНИЕ И ЗАПОЛНЕНИЕ В БОКСЕ: 'TOP', 'RIGHT', 'BOTTOM', 'LEFT'**

Элемент называется *позиционированным*, если его свойство ['position'](#) имеет значение, отличное от 'static'. Позиционированные элементы генерируют позиционированные боксы, располагающиеся в соответствии с четырьмя свойствами:

#### **'top'**

Значение: [<length>](#) | [<percentage>](#) | auto | [inherit](#)  
 Начальное: auto  
 Применяется: к позиционируемым элементам  
 Наследуется: нет  
 Процентное: относительно высоты содержащего блока  
 Носитель: [визуальный](#)

Это свойство специфицирует, насколько смещён вниз верхний край содержимого бокса относительно верхнего края [содержащего блока](#).

#### **'right'**

Значение: [<length>](#) | [<percentage>](#) | auto | [inherit](#)  
 Начальное: auto  
 Применяется: к позиционируемым элементам  
 Наследуется: нет  
 Процентное: относительно ширины содержащего блока  
 Носитель: [визуальный](#)

Это свойство специфицирует, насколько смещён влево правый край содержимого бокса относительно правого края [содержащего блока](#).

**'bottom'**

Значение:	<a href="#">&lt;length&gt;</a>   <a href="#">&lt;percentage&gt;</a>   auto   <a href="#">inherit</a>
Начальное:	auto
Применяется:	к позиционируемым элементам
Наследуется:	нет
Процентное:	относительно высоты содержащего блока
Носитель:	<a href="#">визуальный</a>

Это свойство специфицирует, насколько смещён вверх нижний край содержимого блока относительно низа [содержащего блока](#).

**'left'**

Значение:	<a href="#">&lt;length&gt;</a>   <a href="#">&lt;percentage&gt;</a>   auto   <a href="#">inherit</a>
Начальное:	auto
Применяется:	к позиционируемым элементам
Наследуется:	нет
Процентное:	относительно ширины содержащего блока
Носитель:	<a href="#">визуальный</a>

Это свойство специфицирует, насколько смещён вправо левый край содержимого блока относительно левого края [содержащего блока](#).

Значения для этих четырёх свойств имеют следующий смысл:

**[<length>](#)**

Смещением является расстояние от соответствующего края.

**[<percentage>](#)**

Смещением является процент ширины содержащего блока (для ['left'](#) или ['right'](#)) или высоты содержащего блока (для ['top'](#) и ['bottom'](#)). Для ['top'](#) и ['bottom'](#), если высота содержащего блока не специфицирована явно (т.е. зависит от высоты содержимого), процентное значение интерпретируется так же, как ['auto'](#).

**auto**

Действие этого значения зависит от того, какое из соответствующих свойств также имеет значение ['auto'](#).

См. также разделы [ширина](#) и [высота абсолютно позиционируемых](#), незаменяемых элементов.

Для [абсолютно позиционируемых](#) боксов — смещения являются относительными к [содержащему блоку](#) бокса. Для относительно позиционируемых боксов — смещения являются относительными к внешним краям самого бокса (т.е. боксу задаётся нормальное расположение, а затем — смещение от этой позиции в соответствии с вышеуказанными свойствами).

**НОРМАЛЬНОЕ РАСПОЛОЖЕНИЕ**

Боксы при нормальном обтекании относятся к контексту форматирования, который может быть уровня блока и инлайн, но не тем и другим одновременно. Боксы [блока](#) находятся в контексте [форматирования блока](#). [Инлайн-боксы](#) находятся в контексте [инлайн-форматирования](#).

**КОНТЕКСТ ФОРМАТИРОВАНИЯ БЛОКА**

В контексте форматирования блока боксы устанавливаются один за другим, вертикально, начиная от верха содержащего блока. Вертикальное расстояние между двумя родственными боксами определяется свойством ['margin'](#). Горизонтальные поля между смежными боксами блока в контексте форматирования блока [сжимаются](#).

В контексте форматирования блока левый внешний край бокса касается левого края содержащего блока (для форматирования справа-налево — касается правый край). Это верно даже в случае с поправками (хотя область *содержимого* может усекаться из-за поправок).

Информацию о разрывах страниц в страничных носителях см. в разделе [допустимые разрывы страниц](#).

**КОНТЕКСТ ИНЛАЙН-ФОРМАТИРОВАНИЯ**

В контексте инлайн-форматирования боксы устанавливаются по горизонтали, один за другим, начиная от верха содержащего блока. Горизонтальные поля, рамки и заполнение рассматриваются как отношения между боксами. Боксы могут быть выровнены по вертикали несколькими способами: могут быть выровнены их нижние или верхние края или базовые линии текста внутри них. Прямоугольная область, содержащая боксы, которые образуют строку, называется *строчный бокс*.

Ширина строчного бокса определяется [содержащим блоком](#). Высота строчного бокса определяется правилами из раздела [вычисление высоты строки](#). Строчный бокс всегда имеет высоту, достаточную для содержащихся в нём боксов. В то же время, он может быть выше, чем самый высокий из содержащихся в нём боксов (если, например, боксы выровнены так, что базовые линии выстроены). Если высота бокса В меньше, чем высота строчного бокса, содержащего его, то вертикальное выравнивание В внутри строчного бокса определяется свойством ['vertical-align'](#).

Если несколько инлайн-боксов не входят по горизонтали в один строчный бокс, они распределяются на два или более вертикально упакованных строчных бокса. Таким образом, параграф будет вертикальным стеком из строчных боксов. Строчные боксы упакованы по вертикали без разделения и никогда не перекрываются.

Вообще, левый край строчного бокса касается левого края его содержащего блока, и правый край касается правого края его содержащего блока. В то же время, боксы-поплавки могут появляться между краем содержащего блока и краем строчного блока. Таким образом, хотя строчные боксы в том же самом контексте инлайн-форматирования обычно имеют ту же самую ширину (что и содержащий блок), они могут иметь и другую ширину из-за [поплавков](#), уменьшающих горизонтальное пространство. Строчные боксы в том же самом контексте инлайн-форматирования обычно различаются по высоте (например, одна строка может содержать высокое изображение, а другие строки — только текст).

Если суммарная ширина инлайн-боксов в строке меньше, чем ширина строчного бокса, содержащего их, то их распределение по горизонтали внутри строчного бокса определяется свойством ['text-align'](#). Если это свойство имеет значение 'justify', ПА может уплотнить инлайн-боксы.

Поскольку инлайн-бокс не может превысить ширину строчного бокса, длинные инлайн-боксы разделяются на несколько боксов, и эти боксы распределяются на несколько строчных боксов. Если инлайн-бокс разделён, то поля, рамки и заполнение не имеют визуального эффекта в тех местах, где происходит разделение. Форматирование полей, рамок и заполнения может не быть определено полностью, если разделение происходит внутри двунаправленного внедрения.

Инлайн-боксы могут также быть разделены на несколько боксов *внутри одного строчного бокса* из-за [двунаправленной обработки текста](#).

Вот пример конструкции инлайн-боксов. Следующий параграф (созданный элементом Р уровня блока в HTML) содержит анонимный текст, распределённый между элементами EM и STRONG:

```
<P>Several <EM>emphasized words</EM> appear  
<STRONG>in this</STRONG> sentence, dear.</P>
```

Элемент Р генерирует бокс блока, содержащий пять инлайн-боксов, три из которых — анонимные:

- Anonymous: "Several"
- EM: "emphasized words"
- Anonymous: "appear"
- STRONG: "in this"
- Anonymous: "sentence, dear."

Чтобы сформировать параграф, ПА вставляет пять боксов в строчный бокс. В этом примере бокс, генерируемый для элемента Р, устанавливает содержащий блок для строчных боксов. Если содержащий блок достаточно широк, все инлайн-боксы войдут в один строчный бокс:

```
Several emphasized words appear in this sentence, dear.
```

если нет, инлайн-боксы будут разделены и распределены по нескольким строчным боксам. Предыдущий параграф может быть разделён так:

```
Several emphasized words appear  
in this sentence, dear.
```

или так:

```
Several emphasized  
words appear in this  
sentence, dear.
```

В предыдущем примере бокс EM был разделён на два бокса EM (назовём их "split1" и "split2"). Поля, рамки, заполнение или текстовый орнамент не имеют видимого эффекта после split1 или до split2.

Рассмотри следующий пример:

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
  <HEAD>
    <TITLE>Example of inline flow on several lines</TITLE>
    <STYLE type="text/css">
      EM {
        padding: 2px;
        margin: 1em;
        border-width: medium;
        border-style: dashed;
        line-height: 2.4em;
      }
    </STYLE>
  </HEAD>
  <BODY>
    <P>Several <EM>emphasized words</EM> appear here.</P>
  </BODY>
</HTML>

```

В зависимости от ширины P, боксы могут распределиться так:

- Поле вставлено до "emphasized" и после "words".
- Заполнение вставлено до, сверху и снизу от "emphasized" и после, сверху и снизу от "words". Пунктирная рамка отображается с трёх сторон в каждом случае.

## ОТНОСИТЕЛЬНОЕ ПОЗИЦИОНИРОВАНИЕ

После того, как бокс расположен в соответствии с [нормальным расположением](#), он может быть сдвинут относительно этой позиции. Это называется *относительным позиционированием*. Смещение бокса (B1) таким способом не окажет воздействия на бокс (B2), следующий за ним: B2 получит такую позицию, как будто B1 не смещён и B2 не перемещается после применения смещения B1. Это предполагает, что относительное позиционирование может вызывать взаимное перекрытие боксов.

Относительно позиционированные боксы сохраняют свои нормальные размеры и прорисовку, включая разрывы строк и заполнение, первоначально зарезервированные за ними. Относительно позиционированный бокс устанавливает новый [содержащий блок](#) для нормальной прорисовки дочерних и позиционированных потомков.

Относительно позиционированный бокс генерируется, когда свойство ['position'](#) элемента имеет значение 'relative'. Смещение специфицируется свойствами ['top'](#), ['bottom'](#), ['left'](#) и ['right'](#).

Динамическое перемещение боксов, позиционированных относительно, может создавать эффекты анимации в среде скриптов (см. также свойство ['visibility'](#)). Относительное позиционирование можно использовать также как общую форму под- и надиндексов, за исключением того, что высота строки не устанавливается автоматически при рассмотрении позиционирования. См. дополнительно описание [вычисления высоты строки](#).

Примеры относительного позиционирования есть в разделе [Сравнение нормального позиционирования, поплавок и абсолютного позиционирования](#).

## ПОПЛАВКИ/FLOATS

Поплавки — это боксы, который "всплывает" (смещается) влево или вправо на текущей строке. Самое интересное в поведении поплавок (или "всплывающего", или "плавающего" бокса) то, что содержимое может обтекать его по сторонам (или это может быть запрещено свойством ['clear'](#)). Содержимое обтекает справа бокс, всплывающий влево, и слева — бокс всплывающий вправо.

Далее следует введение в позиционирование поплавок и обтекание содержимого; точные [правила](#), управляющие поведением поплавок, даны в описании свойства ['float'](#).

Всплывающий бокс обязан иметь явно установленную ширину (назначенную свойством ['width'](#), или свою [внутреннюю](#) ширину — в случае [замещаемых элементов](#)). Любой всплывающий бокс становится [боксом блока](#), который сдвигается влево или вправо, пока его внешний край не коснётся края содержащего блока или внешнего края другого поплавок. Верх всплывающего бокса выравнивается с верхом текущего строчного бокса (или низом предшествующего бокса блока, если отсутствует строчный бокс). Если для поплавок не хватает горизонтального пространства на текущей строке, он сдвигается вниз строка за строкой, пока строка не получит достаточно пространства для него.

Пока поплавок не всплыл, непозиционированные боксы блока, созданные до и после всплывающего бокса,



всплывают вертикально, как будто поплавок не существует. Однако строчные боксы, созданные сразу после поплавка, уменьшаются, чтобы дать пространство для всплывающего бокса. Любое содержимое на текущей строке до всплывающего бокса перерисовывается на первой доступной строке с противоположной стороны поплавка. Несколько поплавков могут быть смежными, и эта модель применяется также к смежным поплавкам на той же строке.

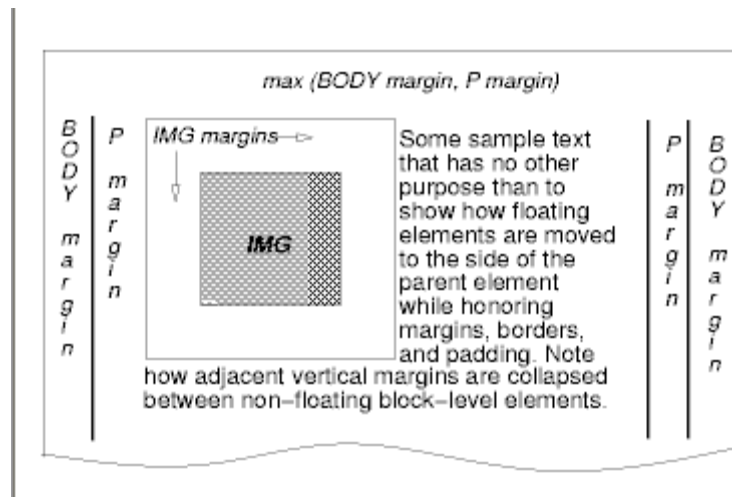
Следующее правило вызывает всплывание влево всех боксов `IMG` с `class="icon"` (и устанавливает левое поле в '0'):

```
IMG.icon {
  float: left;
  margin-left: 0;
}
```

Рассмотрим следующие HTML и таблицу стилей:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
  <HEAD>
    <TITLE>Float example</TITLE>
    <STYLE type="text/css">
      IMG { float: left }
      BODY, P, IMG { margin: 2em }
    </STYLE>
  </HEAD>
  <BODY>
    <P><IMG src=img.gif alt="This image will illustrate floats">
      Some sample text that has no other...
    </P>
  </BODY>
</HTML>
```

`IMG`-бокс всплывает влево. Последующее содержимое форматируется справа от поплавка, начиная с той же строки, где находится поплавок. Строчные боксы справа от поплавка укорачиваются из-за присутствия поплавка, но имеют свою "нормальную" ширину (как в содержащем блоке, установленном элементом `P`) после поплавка. Этот документ может быть сформатирован так:



Форматирование могло бы быть точно таким же, если бы документ был следующим:

```
<BODY>
  <P>Some sample text
  <IMG src=img.gif alt="This image will illustrate floats">
    that has no other...
</BODY>
```

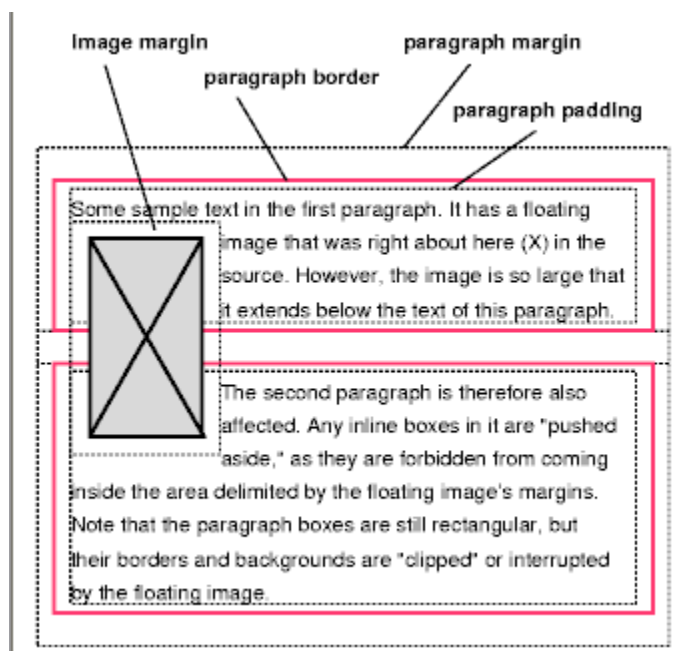
поскольку содержимое слева от поплавка замещается поплавком и обтекает его вниз по его правой стороне.

Поля всплывающих боксов никогда не сжимаются полями смежных боксов. Таким образом, в предыдущем примере вертикальные поля не сжимаются между боксом `P` и всплывающим `IMG`-боксом.

Поплавков может перекрывать другие боксы при нормальном позиционировании (например, когда нормально позиционированный бокс, следующий за поплавком, имеет отрицательные поля). Когда инлайн-бокс перекрывается поплавком, содержимое, фон и рамки инлайн-бокса отображаются впереди поплавка. Если блок бокса перекрывается, фон и рамки бокса блока отображаются позади поплавка и видны только там, где бокс прозрачен. Содержимое бокса блока отображается спереди от поплавка.



Вот другая иллюстрация, показывающая, что происходит, когда поплавков перекрывает рамки элемента с нормальным всплыванием.



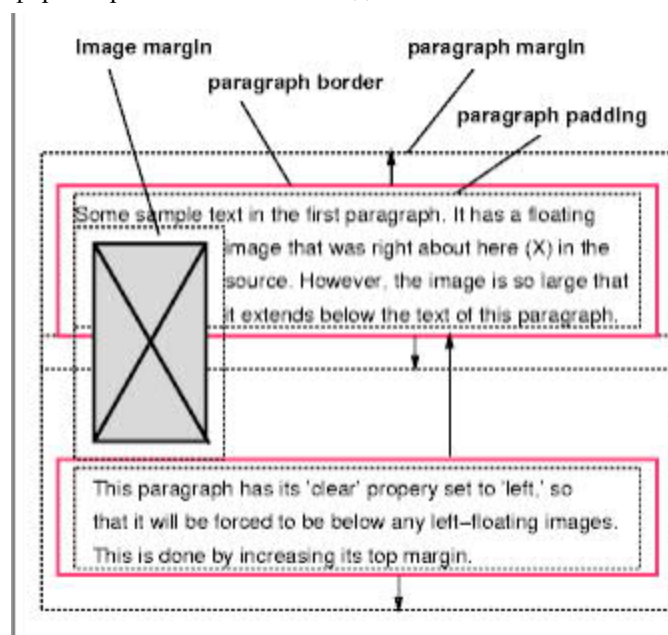
Всплывающее изображение скрывает рамки перекрываемого блока блока.

Следующий пример иллюстрирует использование свойства `'clear'` для предотвращения всплывания содержимого после поплавка.

При таком правиле:

```
P { clear: left }
```

форматирование может выглядеть так:



Оба параграфа установлены в `'clear: left'`, что "выталкивает вниз" второй параграф в положение под поплавком — его верхнее поле растягивается, чтобы выполнить это (см. свойство `'clear'`).

## ПОЗИЦИОНИРОВАНИЕ ПОПЛАВКА: СВОЙСТВО 'FLOAT'

**'float'**

Значение: left | right | none | [inherit](#)

Начальное: none

Применяется: ко всему, кроме позиционированных элементов и генерируемого содержимого

Наследуется: нет

Процентное: N/A

Носитель: [визуальный](#)

Это свойство специфицирует, должен ли бокс всплывать влево, вправо или не всплывать вообще. Оно может быть установлено для элементов, генерирующих боксы, которые не [позиционированы абсолютно](#). Значения этого свойства имеют следующий смысл:

#### **left**

Элемент генерирует [букс блока](#), всплывающий влево. Содержимое обтекает по правой стороне бокса, начиная от верха (субъект свойства ['clear'](#)). 'display' игнорируется, если только оно не имеет значения 'none'.

#### **right**

То же, что 'left', но содержимое обтекает по левой стороне бокса, начиная от верха.

#### **none**

Бокс не всплывает.

Вот точные правила, управляющие поведением поплавок:

1. Левый [внешний край](#) всплывающего влево бокса не может быть слева от левого края его [содержащего блока](#). Аналогичное правило действует для элементов, всплывающих вправо.
2. Если текущий бокс всплывает влево и имеются другие всплывающие влево боксы, сгенерированные элементами в документе-источнике ранее, тогда для каждого такого предшествующего бокса или левый [внешний край](#) текущего бокса обязан быть справа от правого [внешнего края](#) предшествующего бокса, или его верх обязан быть ниже, чем низ предшествующего бокса. Аналогичное правило действует для боксов, всплывающих вправо.
3. Правый [внешний край](#) всплывающего влево бокса не может быть справа от левого [внешнего края](#) какого-либо всплывающего вправо бокса, находящегося справа от него. Аналогичные правила действуют для элементов, всплывающих вправо.
4. [Верхний внешний край](#) всплывающего бокса не может быть выше, чем верх его [содержащего блока](#).
5. [Верхний внешний край](#) всплывающего бокса не может быть выше, чем верхний внешний край любого [всплывающего](#) бокса [блока](#), сгенерированного элементом ранее в документе-источнике.
6. [Верхний внешний край](#) всплывающего бокса элемента не может быть выше, чем верх любого [строчного бокса](#), содержащего бокс, сгенерированный элементом ранее в документе-источнике.
7. Всплывающий влево бокс, имеющий слева от себя другой всплывающий влево бокс, не может иметь свой правый внешний край справа от правого края своего содержащего блока. (Говоря шире: левый поплавок не может находиться у правого края, если только он уже не находится слева так далеко, насколько это возможно.) Аналогичное правило действует для элементов, всплывающих вправо.
8. Всплывающий бокс должен быть размещён так высоко, насколько это возможно.
9. Всплывающий влево бокс обязан быть размещён так далеко влево, насколько это возможно, а всплывающий вправо бокс обязан быть размещён так далеко вправо, насколько это возможно. Более высокая позиция имеет преимущество перед позицией, идущей далее влево/вправо.

## УПРАВЛЕНИЕ ПОЛОЖЕНИЕМ ПОСЛЕ ПОПЛАВКА: СВОЙСТВО 'CLEAR'

### 'clear'

Значение: none | left | right | both | [inherit](#)

Начальное: none

Применяется: к элементам уровня блока

Наследуется: нет

Процентное: N/A

Носитель: [визуальный](#)

Это свойство указывает, какие стороны бокса(ов) элемента *не* могут быть смежными с ранее всплывшим боксом. (Может быть такое, что элемент сам имеет всплывающих потомков; свойство ['clear'](#) не оказывает на них воздействия.)

Это свойство может быть специфицировано только для элементов [уровня блока](#) (включая полавки). Для [компактных](#) и [втягивающихся боксов](#) это свойство применяется к окончательному боксу блока, к которому

компактный или втягивающийся бокс принадлежит.

Значения имеют следующий смысл при применении к невсплывающим боксам блока:

#### **left**

Верхнее поле генерируемого бокса увеличивается настолько, чтобы верхний край рамки был ниже нижнего внешнего края любого всплывающего влево бокса, являющегося результатом действия предыдущих элементов в документе-источнике.

#### **right**

Верхнее поле генерируемого бокса увеличивается настолько, чтобы верхний край рамки был ниже нижнего внешнего края любого всплывающего вправо бокса, являющегося результатом действия предыдущих элементов в документе-источнике.

#### **both**

Генерируемый бокс перемещается ниже всех всплывающих боксов элементов, появившихся ранее в документе-источнике.

#### **none**

Нет ограничений на позицию бокса относительно поплавок.

Если свойство установлено на всплывающие элементы, результатом будет модификация [правил](#) позиционирования поплавка. Прибавляется дополнительное условие (#10):

- [Верхний внешний край](#) поплавка обязан быть ниже нижнего внешнего края всех ранее всплывающих влево боксов (если 'clear: left'), или всех ранее всплывающих вправо боксов (если 'clear: right'), или обоих ('clear: both').

## **АБСОЛЮТНОЕ ПОЗИЦИОНИРОВАНИЕ**

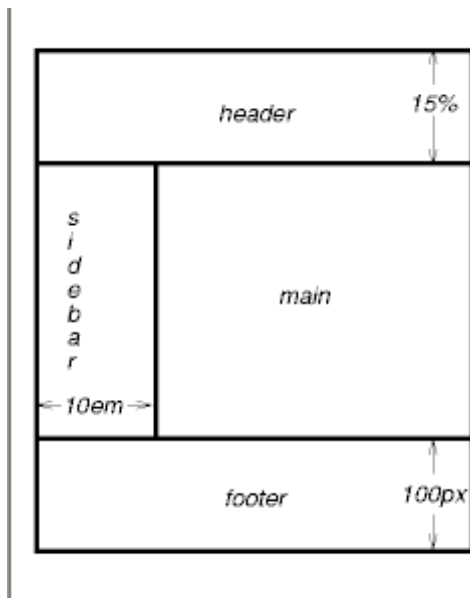
В модели абсолютного позиционирования бокс смещается явно относительно своего содержащего блока. Он полностью удаляется из нормального позиционирования (не влияет на последующих родственников). Абсолютно позиционированный бокс устанавливает новый содержащий блок для нормально позиционируемых дочерних боксов и позиционируемых потомков. В то же время, содержимое абсолютно позиционированных элементов не всплывает вокруг других боксов. Эти элементы могут или могут не закрывать собой содержимое другого бокса, в зависимости от [уровней в стэке](#) у перекрываемых боксов.

Ссылки в данной спецификации на *абсолютно позиционированный элемент* (или его бокс) подразумевают, что свойство ['position'](#) элемента имеет значение 'absolute' или 'fixed'.

## **ФИКСИРОВАННОЕ ПОЗИЦИОНИРОВАНИЕ**

Фиксированное позиционирование это подкатегория абсолютного позиционирования. Единственное отличие в том, что для фиксированно позиционированного бокса содержащий блок устанавливается [портом просмотра](#). Для [непрерывных носителей](#) фиксированные боксы не перемещаются при прокрутке документа. В этом смысле они похожи на [фиксированные фоновые изображения](#). Для [страничных носителей](#) боксы с фиксированной позицией повторяются на каждой странице. Это используется для размещения, к примеру, подписи внизу каждой страницы.

Авторы могут использовать фиксированное позиционирование для создания фреймоподобных презентаций. Рассмотрим следующую структуру фреймов:



[\[D\]](#)

Этого можно добиться с помощью такого документа HTML и таблицы стилей:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
  <HEAD>
    <TITLE>Документ с фреймами в CSS2</TITLE>
    <STYLE type="text/css">
      BODY { height: 8.5in } /* Необходим далее для процентной высоты */
      #header {
        position: fixed;
        width: 100%;
        height: 15%;
        top: 0;
        right: 0;
        bottom: auto;
        left: 0;
      }
      #sidebar {
        position: fixed;
        width: 10em;
        height: auto;
        top: 15%;
        right: auto;
        bottom: 100px;
        left: 0;
      }
      #main {
        position: fixed;
        width: auto;
        height: auto;
        top: 15%;
        right: 0;
        bottom: 100px;
        left: 10em;
      }
      #footer {
        position: fixed;
        width: 100%;
        height: 100px;
        top: auto;
        right: 0;
        bottom: 0;
        left: 0;
      }
    </STYLE>
  </HEAD>
  <BODY>
```

```
<DIV id="header"> ... </DIV>
<DIV id="sidebar"> ... </DIV>
<DIV id="main"> ... </DIV>
<DIV id="footer"> ... </DIV>
</BODY>
</HTML>
```

## СООТНОШЕНИЕ МЕЖДУ 'DISPLAY', 'POSITION' И 'FLOAT'

Три свойства, влияющие на генерацию и структуру бокса -- 'display', ['position'](#) и ['float'](#) -- взаимодействуют так:

1. Если 'display' имеет значение 'none', ПА обязаны [игнорировать](#) ['position'](#) и ['float'](#). В этом случае элемент не генерирует бокса.
2. Иначе, если ['position'](#) имеет значение 'absolute' или 'fixed', 'display' установлен в 'block', а ['float'](#) установлен в 'none'. Позиция бокса будет определяться свойствами ['top'](#), ['right'](#), ['bottom'](#) и ['left'](#) и содержащим блоком бокса.
3. Иначе, если ['float'](#) имеет значение не 'none', 'display' установлен в 'block', и бокс всплывает.
4. Иначе, оставшиеся свойства 'display' применяются так, как специфицированы.

**Примечание.** CSS2 не специфицирует поведение структуры, если значения этих свойств изменяются скриптами. Например, что произойдёт, если элемент, имеющий 'width: auto' изменит позицию? Будет ли перерисовано содержимое или форматирование останется первоначальным? Ответ находится за пределами данного документа, и похоже, что такое поведение отличалось для ранних реализаций CSS2.

## СРАВНЕНИЕ НОРМАЛЬНОГО РАСПОЛОЖЕНИЯ, ПОПЛАВКОВ И АБСОЛЮТНОГО ПОЗИЦИОНИРОВАНИЯ

Чтобы проиллюстрировать разницу между нормальным и относительным позиционированием, поплавками и абсолютным позиционированием, мы предлагаем серию примеров на базе следующего фрагмента HTML:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
  <HEAD>
    <TITLE>Сравнение схем позиционирования</TITLE>
  </HEAD>
  <BODY>
    <P>Beginning of body contents.
      <SPAN id="outer"> Start of outer contents.
        <SPAN id="inner"> Inner contents.</SPAN>
      End of outer contents.</SPAN>
    End of body contents.
  </P>
</BODY>
</HTML>
```

Для этого документа мы принимаем следующие правила:

```
BODY { display: block; line-height: 200%;
       width: 400px; height: 400px }
P     { display: block }
SPAN  { display: inline }
```

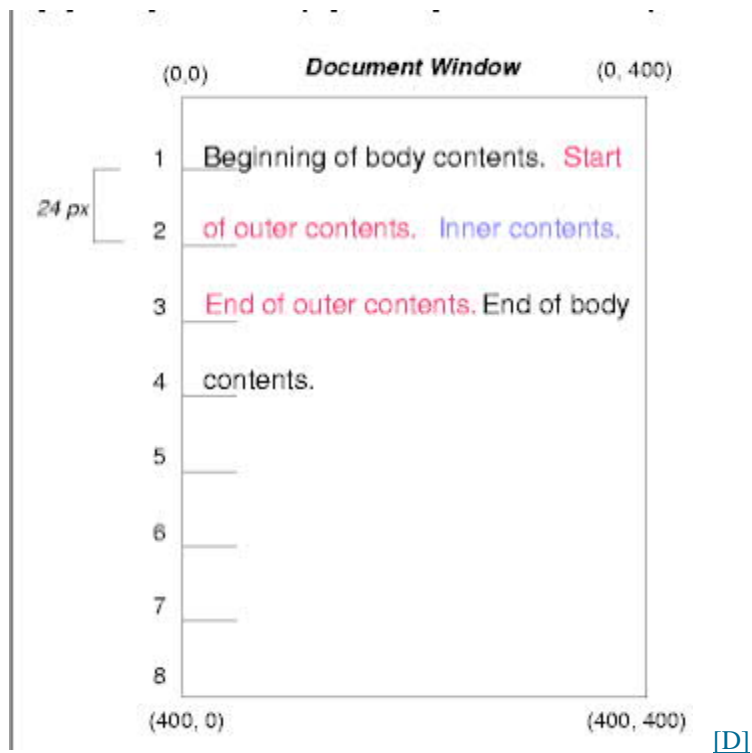
Окончательные позиции боксов, генерируемых *внешними* и *внутренними* элементами, различны в каждом примере. В каждой иллюстрации числа слева от иллюстрации обозначают позицию [нормального позиционирования](#) строк. (Примечание: иллюстрации используют различные горизонтальные и вертикальные масштабы.)

## НОРМАЛЬНОЕ РАСПОЛОЖЕНИЕ

Рассмотрим следующие объявления CSS для *outer* и *inner*, которые не изменяют [нормального расположения](#) боксов:

```
#outer { color: red }
#inner { color: blue }
```

Элемент P содержит всё инлайн-содержимое: [анонимный инлайн-текст](#) и два элемента SPAN. Следовательно, всё содержимое будет расположено вне инлайн-контекста форматирования внутри содержащего блока, устанавливаемого элементом P, и получится что-то подобное:



## ОТНОСИТЕЛЬНОЕ ПОЗИЦИОНИРОВАНИЕ

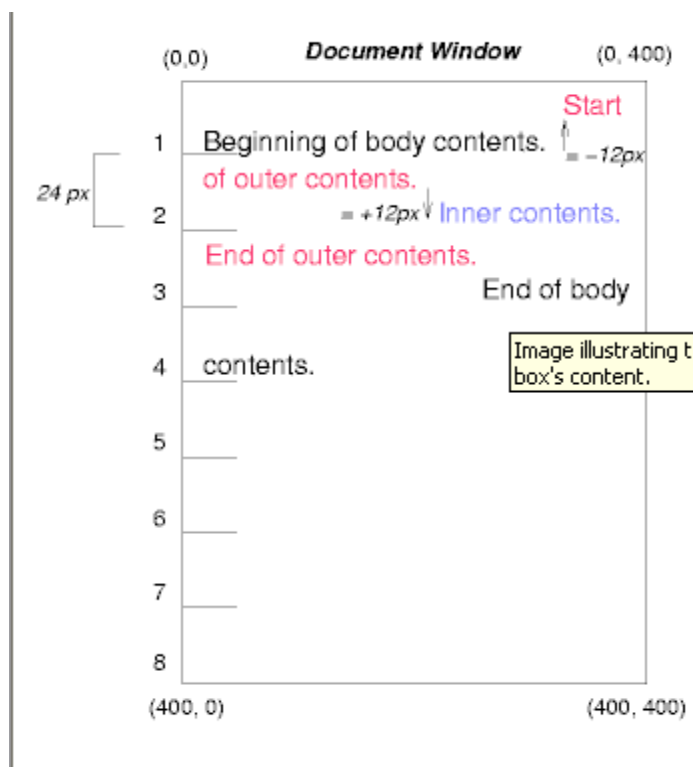
Чтобы увидеть действие [относительного позиционирования](#), мы специфицируем:

```
#outer { position: relative; top: -12px; color: red }
#inner { position: relative; top: 12px; color: blue }
```

Текст расположен нормально до элемента *outer*. Текст *outer* затем всплывает в своей нормальной позиции и размерах в конце строки 1. Затем инлайн-боксы, содержащие текст (распределённый на три строки), сдвигаются вместе на '-12px' (вверх).

Содержимое *inner* как дочернего от *outer* было бы нормально расположено сразу после слов "of outer contents" (на строке 1.5). Однако содержимое *inner* само смещено относительно содержимого *outer* на '12px' (вниз), обратно на свою первоначальную позицию на строке 2.

Обратите внимание, что на содержимое, следующее после *outer*, относительное позиционирование *outer* не влияет.



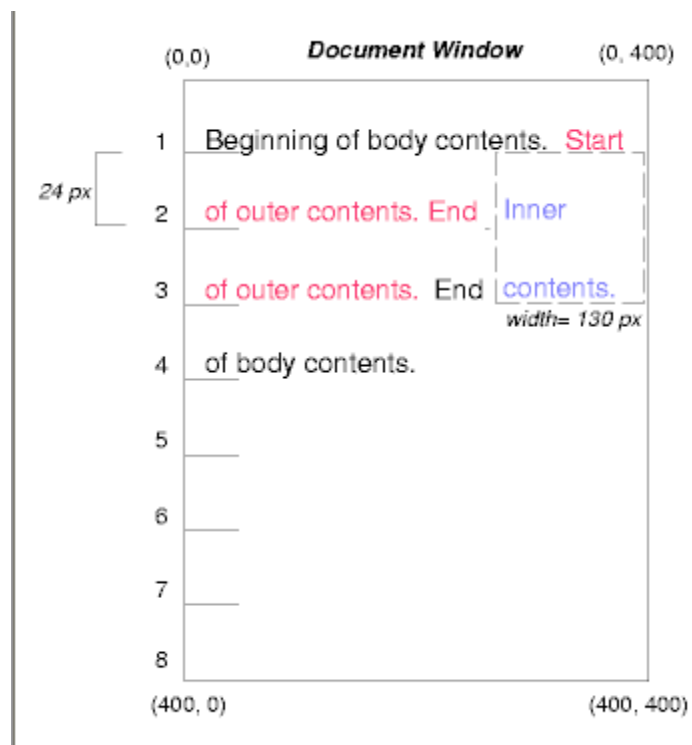
Заметьте также, что, имея смещение для *outer* '-24px', текст *outer* и текст тела могут быть перекрыты.

## ОБТЕКАНИЕ БОКСА

Теперь рассмотрим эффект от [всплывания](#) текста элемента *inner* вправо при использовании следующих правил:

```
#outer { color: red }
#inner { float: right; width: 130px; color: blue }
```

Текст нормально расположен до бокса *inner*, который вытолкнут из расположения и всплыл к правому полю (его ['width'](#) установлено явно). Строчные боксы слева от поплавка укорочены, и оставшийся текст документа всплывает в них.



[D]

Чтобы увидеть действие свойства ['clear'](#), мы добавим в пример *родственные* элементы:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
  <HEAD>
    <TITLE>Сравнение схем позиционирования II</TITLE>
  </HEAD>
  <BODY>
    <P>Beginning of body contents.
      <SPAN id=outer> Start of outer contents.
        <SPAN id=inner> Inner contents.</SPAN>
        <SPAN id=sibling> Sibling contents.</SPAN>
        End of outer contents.</SPAN>
        End of body contents.
    </P>
  </BODY>
</HTML>
```

Следующие правила:

```
#inner { float: right; width: 130px; color: blue }
#sibling { color: red }
```

заставляют бокс *inner* всплывать вправо, как и раньше, а остальной текст документа — всплывать в незанятом пространстве:

Однако, если свойство ['clear'](#) *родственного* элемента установлено в 'right' (т.е. генерируемый *родственный* бокс не займёт позицию справа от всплывающего бокса), *родственное* содержимое начнёт располагаться ниже поплавка:

```
#inner { float: right; width: 130px; color: blue }
#sibling { clear: right; color: red }
```



## АБСОЛЮТНОЕ ПОЗИЦИОНИРОВАНИЕ

Наконец, мы рассмотрим действие [абсолютного позиционирования](#). Вот объявления CSS для *outer* и *inner*:

```
#outer {
  position: absolute;
  top: 200px; left: 200px;
  width: 200px;
  color: red;
}
#inner { color: blue }
```

которые позиционируют верх бокса *outer* относительно его содержащего блока. Содержащий блок для позиционируемого бокса устанавливается ближайшим позиционированным предком (или, если таковой не существует, [начальным содержащим блоком](#), как в нашем примере). Верхняя сторона бокса *outer* на '200px' ниже верха содержащего блока, а левая сторона — на '200px' от левой стороны. Дочерний бокс *outer*'а позиционирован нормально относительно родителя.

Следующий пример покажет абсолютно позиционированный бокс, являющийся дочерним от относительно позиционированного бокса. Хотя родительский бокс *outer* не имеет смещения, установка его свойства ['position'](#) в *'relative'* означает, что этот бокс может служить как содержащий блок для позиционированных потомков. Поскольку бокс *outer* это инлайн-бокс, разделённый на несколько строк, край верха первого инлайн-бокса и левый край (обозначенный толстой пунктирной линией на рисунке ниже) служат как ссылки для смещений *'top'* и *'left'*.

```
#outer {
  position: relative;
  color: red
}
#inner {
  position: absolute;
  top: 200px; left: -100px;
  height: 130px; width: 130px;
  color: blue;
}
```

В результате получится примерно так:

Если мы не позиционируем бокс *outer*:

```
#outer { color: red }
#inner {
  position: absolute;
  top: 200px; left: -100px;
  height: 130px; width: 130px;
  color: blue;
}
```

содержащий блок для *inner* станет [начальным содержащим блоком](#) (в нашем примере). Следующая иллюстрация показывает, где в этом случае окажется бокс *inner*.

Относительное и абсолютное позиционирование может использоваться для выполнения смены баннеров, как показано в следующем примере. Такой документ:

```
<P style="position: relative; margin-right: 10px; left: 10px;">
I used two red hyphens to serve as a change bar. They
will "float" to the left of the line containing THIS
<SPAN style="position: absolute; top: auto; left: -1em; color: red;">--</SPAN>
word.</P>
```

может дать в результате примерно это:

Сначала параграф (стороны содержащего блока которого показаны на иллюстрации) расположен нормально. Затем он смещается на '10px' от левого края содержащего блока (таким образом, правое поле в '10px' резервируется для предполагаемого смещения). Два дефиса, работающие как сменяющиеся баннеры, изымаются из расположения и позиционируются на текущей строке (поскольку *'top: auto'*), на *'-1em'* от левого края своего содержащего блока (установленного *P* в свою окончательную позицию). В результате сменяющиеся баннеры "всплывают" слева от текущей строки.

## СЛОИ

В последующих разделах выражение *"спереди от"* означает ближе к пользователю, смотрящему на экран.

В CSS2 каждый бокс имеет позицию в трёх измерениях. В дополнение к позиции относительно вертикали и

горизонтали, боксы расположены вдоль "z-axis/оси z" и форматируются один над другим. Позиции по оси z обычно рассматриваются, когда боксы перекрываются визуально. В этом разделе обсуждается, как боксы можно позиционировать относительно оси z.

Каждый бокс принадлежит к *контексту стэка*. Каждый бокс в данном контексте стэка имеет целочисленный *уровень стэка*, являющийся позицией бокса по оси z относительно других боксов того же самого контекста стэка. Боксы с большим уровнем стэка всегда форматируются перед боксами с меньшими уровнями стэка. Боксы могут иметь отрицательные значения уровня стэка. Боксы, имеющие тот же уровень в контексте стэка, упакованы снизу-вверх в соответствии с порядком дерева документа.

[Корневой](#) элемент создаёт *корневой контекст стэка*, но другие элементы могут устанавливать *локальные контексты стэка*. Контексты стэка наследуются. Локальный контекст стэка первичен; боксы других контекстов стэка не могут появляться между его боксами.

Элемент, устанавливающий локальный контекст стэка, генерирует бокс, который имеет два уровня стэка: один для контекста создаваемого стэка (всегда '0') и второй для контекста стэка, к которому он (бокс) принадлежит (задаваемый свойством '[z-index](#)').

Бокс элемента имеет тот же уровень стэка, что и его бокс-родитель, если только не задан другой уровень стэка свойством '[z-index](#)'.

## СПЕЦИФИКАЦИЯ УРОВНЯ В ПАКЕТЕ СЛОЁВ: СВОЙСТВО 'Z-INDEX'

### 'z-index'

Значение:	auto   <a href="#">&lt;integer&gt;</a>   <a href="#">inherit</a>
Начальное:	auto
Применяется:	к позиционированным элементам
Наследуется:	нет
Процентное:	N/A
Носитель:	<a href="#">визуальный</a>

Для позиционированных боксов свойство '[z-index](#)' специфицирует:

1. Уровень стэка бокса в текущем контексте стэка.
2. Устанавливает ли бокс локальный контекст стэка.

Значения имеют следующий смысл:

#### [<integer>](#)

Это целое число — уровень стэка сгенерированного бокса в текущем контексте стэка. Бокс также устанавливает локальный контекст стэка со своим уровнем в стэке '0'.

#### auto

Уровень стэка сгенерированного бокса в текущем контексте стэка тот же, что и у бокса-родителя. Бокс не устанавливает новый локальный контекст стэка.

В следующем примере уровни стэка боксов (именованных своими атрибутами "id"): "text2"=0, "image"=1, "text3"=2 и "text1"=3. Уровень стэка для "text2" наследуется от корневого бокса. Остальные — специфицируются свойством '[z-index](#)'.

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
  <HEAD>
    <TITLE>Z-позиционирование</TITLE>
    <STYLE type="text/css">
      .pile {
        position: absolute;
        left: 2in;
        top: 2in;
        width: 3in;
        height: 3in;
      }
    </STYLE>
  </HEAD>
  <BODY>
    <P>
      <IMG id="image" class="pile"

```

```

        src="butterfly.gif" alt="A butterfly image"
        style="z-index: 1">

<DIV id="text1" class="pile"
    style="z-index: 3">
    This text will overlay the butterfly image.
</DIV>

<DIV id="text2">
    This text will be beneath everything.
</DIV>

<DIV id="text3" class="pile"
    style="z-index: 2">
    This text will underlay text1, but overlay the butterfly image
</DIV>
</BODY>
</HTML>

```

Этот пример демонстрирует понятие *прозрачности*. Поведение по умолчанию бокса — позволять боксам, находящимся сзади, быть видимыми сквозь прозрачные области своего содержимого. В примере каждый бокс прозрачно накладывается на боксы ниже себя. Это поведение может быть переопределено путём использования одного из существующих [свойств фона](#).

## НАПРАВЛЕНИЕ ТЕКСТА: СВОЙСТВА 'DIRECTION' И 'UNICODE-BIDI'

Символы в некоторых видах письма записываются справа налево. В некоторых документах, особенно на арабском и еврейском, и в некоторых контекстах смешивания языков текст в одиночном (визуально отображаемом) блоке может появляться со смешанным направлением письма. Этот феномен называется *bidirectionality* \двунаправленность, или сокращённо — "bidi".

Стандарт Unicode ([UNICODE](#), раздел 3.11) даёт сложный алгоритм определения направления текста. Этот алгоритм состоит из подразумеваемой части, базирующейся на свойствах символов, а также явной, контролирующей внедрения и переопределения. CSS2 обращается к этому алгоритму для достижения соответствующего двунаправленного отображения. Свойства ['direction'](#) и ['unicode-bidi'](#) позволяют авторам специфицировать, как элементы и атрибуты языка документа отображаются в этот алгоритм.

Если документ содержит символы справа-налево и если ПА отображает эти символы соответствующими глифами (а не произвольно замещающими, такими как знак кавычки, 16-ричный код, чёрный бокс и т.п.), то ПА обязан применять двунаправленный алгоритм. Это кажущееся односторонним требование отражает тот факт, что, хотя не каждый еврейский или арабский документ содержит текст разных направлений, велика вероятность того, что эти документы могут содержать текст слева-направо (например, числа, текст из других языков).

Поскольку направление текста зависит от структуры и семантики документа, эти свойства должны в большинстве случаев использоваться только дизайнерами описания типа документа (ОТД) или авторами специальных документов. Если таблица стилей по умолчанию специфицирует эти свойства, то авторы и пользователи не должны специфицировать правила для их переопределения. Типичным исключением будет переопределение bidi-поведения в ПА, если этот ПА производит транслитерацию идиша (обычно записанного буквами иврита) на латиницу по запросу пользователя.

Спецификация HTML 4.0 ([HTML40], раздел 8.2) определяет двунаправленное поведение для элементов HTML. [Соответствующие](#) HTML ПАгенты могут поэтому игнорировать свойства ['direction'](#) и ['unicode-bidi'](#) в авторских и пользовательских таблицах стилей. Правила таблиц стилей, которые необходимо применять для bidi-поведения и которые специфицированы в [\[HTML40\]](#), даны в [примере таблицы стилей](#). Спецификация HTML 4.0 содержит также дополнительную информацию о вопросах двунаправленности.

### 'direction'

Значение:	ltr   rtl   <a href="#">inherit</a>
Начальное:	ltr
Применяется:	ко всем элементам, но см. текст
Наследуется:	да
Процентное:	N/A
Носитель:	<a href="#">визуальный</a>

Это свойство специфицирует базовое направление письма блоков и направление внедрений и переопределений

(см. ['unicode-bidi'](#)) для двунаправленного алгоритма Unicode. Дополнительно оно специфицирует направление для столбца [таблицы](#), направление горизонтального [переполнения](#) и позицию неполной последней строки блока в том случае, если `'text-align: justify'`.

Значения этого свойства имеют следующий смысл:

#### **ltr**

Направление слева направо.

#### **rtl**

Справа налево.

Чтобы свойство ['direction'](#) работало в элементах инлайн-уровня, значение свойства ['unicode-bidi'](#) обязано быть `'embed'` или `'override'`.

***Примечание.** Свойство ['direction'](#), специфицированное для элементов столбца таблицы, не наследуется ячейками столбца, поскольку столбцы не существуют в дереве документа. Таким образом, CSS не может использовать правила наследования атрибута `"dir"`, описанные в [\[HTML40\]](#), в разделе 11.3.2.1.*

#### **'unicode-bidi'**

Значение: normal | embed | bidi-override | [inherit](#)

Начальное: normal

Применяется: ко всем элементам, но см. текст

Наследуется: нет

Процентное: N/A

Носитель: [визуальный](#)

Значения этого свойства имеют следующий смысл:

#### **normal**

Элемент не открывает дополнительный уровень внедрения относительно двунаправленного алгоритма. Для элементов инлайн-уровня неявное переупорядочивание работает вне границ элемента.

#### **embed**

Если элемент — инлайн-уровня, это значение открывает дополнительный уровень внедрения относительно двунаправленного алгоритма. Направление в этом уровне внедрения задаётся свойством ['direction'](#). Внутри элемента переупорядочивание выполняется неявно. Это соответствует дополнению LRE (U+202A; для `'direction: ltr'`) или RLE (U+202B; для `'direction: rtl'`) в начале элемента и PDF (U+202C) в конце элемента.

#### **bidi-override**

Если элемент — уровня блока или инлайн и содержит только элементы инлайн-уровня, это значение создаёт переопределение. Это означает, что внутри элемента переупорядочивание выполняется строго в соответствии со свойством ['direction'](#); неявная часть двунаправленного алгоритма игнорируется. Это соответствует дополнению LRO (U+202D; для `'direction: ltr'`) или RLO (U+202E; для `'direction: rtl'`) в начале элемента и а PDF (U+202C) в конце элемента.

Окончательный порядок символов в каждом элементе уровня блока — такой, как если бы `bidi`-код управления был добавлен так, как описано выше, разметка была бы вырезана, а результирующая последовательность символов — передана в двунаправленный алгоритм Unicode в обычный текст, который производил бы те же самые разрывы строк, что и стилизованный текст. В этом процессе нетекстуальные объекты, такие как изображения, рассматриваются как нейтральные символы, если только их свойство ['unicode-bidi'](#) не имеет значений, отличных от `'normal'`, тогда они рассматриваются как полужирные (`strong`) символы в ['direction'](#), специфицированном для элемента.

Пожалуйста, обратите внимание, что, для того чтобы иметь возможность разместить инлайн-боксы в одном направлении (все слева-направо или все справа-налево), может понадобиться создание дополнительных инлайн-боксов (включая анонимные инлайн-боксы), и понадобится разделить и переупорядочить некоторые инлайн-боксы до размещения.

Поскольку алгоритм Unicode имеет предел — 15 уровней внедрения, лучше не использовать ['unicode-bidi'](#) со значениями, отличными от `'normal'`, если отсутствуют подходящие. Значение `'inherit'` должно использоваться особенно осторожно. Однако для элементов, которые обычно предполагается отображать как блоки, установка `'unicode-bidi: embed'` предпочтительнее для удержания элементов вместе в том случае, когда дисплей изменяется на инлайн (см. пример ниже).

В следующем примере показан документ XML с двунаправленным текстом. Он иллюстрирует важный принцип

дизайна: дизайнеры ОТД должны принимать в расчёт bidі и в собственно языке (элементы и атрибуты), и в сопровождающих таблицах стилей. Таблицы стилей должны быть разработаны так, чтобы правила bidі были отделены от других правил стиля. Правила bidі не должны переопределяться другими таблицами стилей, чтобы сохранить поведение bidі языка и ОТД.

## CSS: КУРСОРЫ

Internet Explorer позволяет задавать стили для курсоров. Некоторые стили доступны только для IE 6.

all-scroll	Курсор со стрелками во все четыре стороны и точкой в центре, показывающий на возможность скроллинга страницы в любом направлении I {cursor: all-scroll;}
auto	По умолчанию. Браузер определяет самостоятельно, какой курсор требуется в данном контексте I {cursor: auto;}
col-resize	Курсор со стрелками влево-вправо и вертикальной разделяющей полоской. Используется для индикации возможности изменения размеров по горизонтали H4 {cursor: col-resize;}
crosshair	Курсор-крест H4 {cursor: crosshair;}
default	Стандартный курсор, используемый системой H4 {cursor: default;}
hand	Рука с вытянутым указательным пальцем. Используется при гиперссылке H4 {cursor: hand;}
help	Стрелка с вопросительным знаком. H3 {cursor: help;}
move	Курсор со 4 стрелками, показывающий возможность перемещения H2 {cursor: move;}
no-drop	Рука с перечеркнутым кружочком. Нельзя сбросить объект в текущую позицию курсора TD {cursor: no-drop;}
not-allowed	Перечеркнутый круг. Данная операция не поддерживается TD {cursor: not-allowed;}
pointer	Идентична стилю hand TD {cursor: pointer;}
progress	Песочные часы, показывающие на продолжение операции TD {cursor: progress;}
row-resize	Курсор со стрелками вверх-вниз и вертикальной разделяющей полоской. Используется для индикации возможности изменения размеров по вертикали TD {cursor: row-resize;}
text	Текстовый курсор-каретка TD {cursor: text;}
url(uri)	Ваш собственный курсор. Поддерживаются файлы .cur и .ani TD {cursor:url(elogo.cur);}
vertical-text	Горизонтальная текстовая каретка для вертикального текста TD {cursor: vertical-text;}
wait	Курсор, показывающий, что система занята и требуется подождать TD {cursor: wait;}

*-resize	Курсоры, показывающие возможность потянуть за край окна. Вместо символа * используйте N, NE, NW, S, SE, SW, E, или W, определяющие направление стрелок TD {cursor: n-resize;}
----------	--

### Примеры

Проведите курсоры над ячейками таблицы для демонстрации стилей

all-scroll	auto	col-resize	crosshair	default	hand
help	move	no-drop	not-allowed	pointer	progress
row-resize	text	vertical-text	wait		
n-resize	s-resize	ne-resize	sw-resize		
nw-resize	se-resize	e-resize	w-resize		
url(new.cur)	url(kiss.ani)				

## CSS: ПОЛОСА ПРОКРУТКИ

Данные настройки применимы ко всем элементам, имеющим полосы прокрутки: сама страница (элемент BODY), текстовый блок (TEXTAREA) и т.д.

scrollbar-3dlight-color	Определяет или устанавливает цвет верха и левой части ползунка и кнопок со стрелками на полосе прокрутки body { scrollbar-3dlight-color: green; }
scrollbar-arrow-color	Устанавливает или определяет цвет стрелок на кнопке со стрелками body { scrollbar-arrow-color: red; }
scrollbar-base-color	Устанавливает или определяет цвет основных элементов ползунка: ползунка, кнопок со стрелками, дорожки для ползунка, если не определены параметры в scrollbar-face-color body { scrollbar-base-color: green; }
scrollbar-darkshadow-color	Устанавливает или определяет цвет тени для ползунка и кнопок со стрелками body { scrollbar-darkshadow-color: red; }
scrollbar-face-color	Устанавливает или определяет цвет ползунка и кнопок со стрелками. Также, если вы не задали параметр SCROLLBAR-TRACK-COLOR, у вас изменится цвет дорожки body { scrollbar-face-color: green; }
scrollbar-highlight-color	Устанавливает или получает цвет подсветки, создающий эффект объёмности. Это цвет, который окаймляет освещённую часть кнопочки. Когда кнопка не нажата, то цвет заливает левый верхний угол и стороны между ним, когда нажата — нижний правый угол body { scrollbar-highlight-color: green; }
scrollbar-shadow-color	Схоже с scrollbar-darkshadow-color body { scrollbar-shadow-color: green; }
scrollbar-track-color	Устанавливает или получает цвет дорожки для ползунка body { scrollbar-track-color: aqua; }

## СПЕЦСИМВОЛЫ

Имя	Код	Вид	Описание
&quot;	&#34;	"	двойная кавычка
&amp;	&#38;	&	амперсанд
&lt;	&#60;	<	знак 'меньше'

&gt;	&#62;	>	знак 'больше'
&nbsp;	&#160;		неразрывный пробел
&iexcl;	&#161;	¡	перевернутый восклицательный знак
&cent;	&#162;	¢	цент
&pound;	&#163;	£	фунт стерлингов
&curren;	&#164;	¤	денежная единица
&yen;	&#165;	¥	иена или юань
&brvbar;	&#166;	¦	разорванная вертикальная черта
&sect;	&#167;	§	параграф
&uml;	&#168;	¨	умляут
&copy;	&#169;	©	знак copyright
&ordf;	&#170;	ª	женский порядковый числитель
&laquo;	&#171;	«	левая двойная угловая скобка
&not;	&#172;	¬	знак отрицания
&shy;	&#173;		место возможного переноса
&reg;	&#174;	®	знак зарегистрированной торговой марки
&macr;	&#175;	¯	верхняя горизонтальная черта
&deg;	&#176;	°	градус
&plusmn;	&#177;	±	плюс-минус
&sup2;	&#178;	²	"в квадрате"
&sup3;	&#179;	³	"в кубе"
&acute;	&#180;	´	знак ударения
&micro;	&#181;	µ	микро
&para;	&#182;	¶	символ параграфа
&middot;	&#183;	·	точка
&cedil;	&#184;	¸	седиль (орфографический знак)
&sup1;	&#185;	¹	верхний индекс 'один'
&ordm;	&#186;	º	мужской порядковый числитель
&raquo;	&#187;	»	правая двойная угловая скобка
&frac14;	&#188;	¼	одна четвертая
&frac12;	&#189;	½	одна вторая
&frac34;	&#190;	¾	три четвертых
&iquest;	&#191;	¿	перевернутый вопросительный знак
&Agrave;	&#192;	À	латинская заглавная А с тупым ударением
&Aacute;	&#193;	Á	латинская заглавная А с острым ударением
&Acirc;	&#194;	Â	латинская заглавная А с диакритическим знаком над гласной
&Atilde;	&#195;	Ã	латинская заглавная А с тильдой
&Auml;	&#196;	Ä	латинская заглавная А с двумя точками
&Aring;	&#197;	Å	латинская заглавная А с верхним кружком
&AElig;	&#198;	Æ	латинские заглавные символы АЕ вместе
&Ccedil;	&#199;	Ç	латинская заглавная С с седилом



&Egrave;	&#200;	È	латинская заглавная Е с тупым ударением
&Eacute;	&#201;	É	латинская заглавная Е с острым ударением
&Ecirc;	&#202;	Ê	латинская заглавная Е с диакритическим знаком над гласной
&Euml;	&#203;	Ë	латинская заглавная Е с двумя точками
&Igrave;	&#204;	Ì	латинская заглавная I с тупым ударением
&Iacute;	&#205;	Í	латинская заглавная I с острым ударением
&Icirc;	&#206;	Î	латинская заглавная I с диакритическим знаком над гласной
&Iuml;	&#207;	Ï	латинская заглавная I с двумя точками
&ETH;	&#208;	Ð	латинская заглавная D с черточкой
&Ntilde;	&#209;	Ñ	латинская заглавная N с тильдой
&Ograve;	&#210;	Ò	латинская заглавная O с тупым ударением
&Oacute;	&#211;	Ó	латинская заглавная O с острым ударением
&Ocirc;	&#212;	Ô	латинская заглавная O с диакритическим знаком над гласной
&Otilde;	&#213;	Õ	латинская заглавная O с тильдой
&Ouml;	&#214;	Ö	латинская заглавная O с двумя точками
&times;	&#215;	×	знак умножения
&Oslash;	&#216;	Ø	латинская заглавная O со штрихом
&Ugrave;	&#217;	Ù	латинская заглавная U с тупым ударением
&Uacute;	&#218;	Ú	латинская заглавная U с острым ударением
&Ucirc;	&#219;	Û	латинская заглавная U с диакритическим знаком
&Uuml;	&#220;	Ü	латинская заглавная U с двумя точками
&Yacute;	&#221;	Ý	латинская заглавная Y с острым ударением
&THORN;	&#222;	Þ	латинская заглавная THORN
&agrave;	&#224;	à	латинская строчная а с тупым ударением
&aacute;	&##225;	á	латинская строчная а с острым ударением
&acirc;	&##226;	â	латинская строчная а с диакритическим знаком
&atilde;	&#227;	ã	латинская строчная а с тильдой
&auml;	&#228;	ä	латинская строчная а с двумя точками
&aring;	&#229;	å	латинская строчная а с верхним кружком
&aelig;	&#230;	æ	латинская строчные буквы æ
&ccedil;	&#231;	ç	латинская строчная с с седилем
&egrave;	&#232;	è	латинская строчная е с тупым ударением
&eacute;	&#233;	é	латинская строчная е с острым ударением
&ecirc;	&#234;	ê	латинская строчная е с диакритическим знаком
&euml;	&#235;	ë	латинская строчная е с двумя точками
&igrave;	&#236;	ì	латинская строчная I с тупым ударением
&iacute;	&#237;	í	латинская строчная I с острым ударением
&icirc;	&#238;	î	латинская строчная I с диакритическим знаком
&iuml;	&#239;	ï	латинская строчная I с двумя точками
&eth;	&#240;	ð	латинская строчные символы eth
&ntilde;	&#241;	ñ	латинская строчная N с тильдой

&ograve;	&#242;	ò	латинская строчная О с тупым ударением
&oacute;	&#243;	ó	латинская строчная О с острым ударением
&ocirc;	&#244;	ô	латинская строчная О с диакритическим знаком
&otilde;	&#245;	õ	латинская строчная I с тильдой
&ouml;	&#246;	ö	латинская строчная I с двумя точками
&divide;	&#247;	÷	знак деления
&oslash;	&#248;	ø	латинская строчная О со штрихом
&ugrave;	&#249;	ù	латинская строчная U с тупым ударением
&uacute;	&#250;	ú	латинская строчная U с острым ударением
&ucirc;	&#251;	û	латинская строчная U с диакритическим знаком
&uuml;	&#252;	ü	латинская строчная U с двумя точками
&yacute;	&#253;	ý	латинская строчная Y с острым ударением
&thorn;	&#254;	þ	латинская строчная thorn
&yuml;	&#255;	ÿ	латинская строчная Y с двумя точками
&fnof;	&#402;	f	знак функции
<b>Греческие буквы</b>			
&Alpha;	&#913;	Α	заглавная альфа
&Beta;	&#914;	Β	заглавная бета
&Gamma;	&#915;	Γ	заглавная гамма
&Delta;	&#916;	Δ	заглавная дельта
&Epsilon;	&#917;	Ε	заглавная эпсилон
&Zeta;	&#918;	Ζ	заглавная дзета
&Eta;	&#919;	Η	заглавная эта
&Theta;	&#920;	Θ	заглавная тета
&Iota;	&#921;	Ι	заглавная иота
&Kappa;	&#922;	Κ	заглавная каппа
&Lambda;	&#923;	Λ	заглавная лямбда
&Mu;	&#924;	Μ	заглавная мю
&Nu;	&#925;	Ν	заглавная ню
&Xi;	&#926;	Ξ	заглавная кси
&Omicron;	&#927;	Ο	заглавная омикрон
&Pi;	&#928;	Π	заглавная пи
&Rho;	&#929;	Ρ	заглавная ро
&Sigma;	&#931;	Σ	заглавная сигма
&Tau;	&#932;	Τ	заглавная тау
&Upsilon;	&#933;	Υ	заглавная ипсилон
&Phi;	&#934;	Φ	заглавная фи
&Chi;	&#935;	Χ	заглавная хи
&Psi;	&#936;	Ψ	заглавная пси
&Omega;	&#937;	Ω	заглавная омега
&alpha;	&#945;	α	строчная альфа
&beta;	&#946;	β	строчная бета

&gamma;	&#947;	γ	строчная гамма
&delta;	&#948;	δ	строчная дельта
&epsilon;	&#949;	ε	строчная эпсилон
&zeta;	&#950;	ζ	строчная дзета
&eta;	&#951;	η	строчная эта
&theta;	&#952;	θ	строчная тета
&iota;	&#953;	ι	строчная иота
&kappa;	&#954;	κ	строчная каппа
&lambda;	&#955;	λ	строчная лямбда
&mu;	&#956;	μ	строчная мю
&nu;	&#957;	ν	строчная ню
&xi;	&#958;	ξ	строчная кси
&omicron;	&#959;	ο	строчная омикрон
&pi;	&#960;	π	строчная пи
&rho;	&#961;	ρ	строчная ро
&sigmaf;	&#962;	ς	строчная сигма(final)
&sigma;	&#963;	σ	строчная сигма
&tau;	&#964;	τ	строчная тау
&upsilon;	&#965;	υ	строчная ипсилон
&phi;	&#966;	φ	строчная фи
&chi;	&#967;	χ	строчная хи
&psi;	&#968;	ψ	строчная пси
&omega;	&#969;	ω	строчная омега
<b>Стрелки</b>			
&larr;	&#8592;	←	стрелка влево
&uarr;	&#8593;	↑	стрелка вверх
&rarr;	&#8594;	→	стрелка вправо
&darr;	&#8595;	↓	стрелка вниз
&harr;	&#8596;	↔	стрелка влево-вправо
<b>Прочие символы</b>			
&spades;	&#9824;	♠	знак масти 'пики'
&clubs;	&#9827;	♣	знак масти 'трефы'
&hearts;	&#9829;	♥	знак масти 'червы'
&diams;	&#9830;	♦	знак масти 'бубны'
&circ;	&#710;	ˆ	диакритический знак над гласной
&tilde;	&#732;	˜	тильда
&trade;	&#8482;	™	знак торговой марки
<b>Знаки пунктуации</b>			
&bull;	&#8226;	•	маленький черный кружок
&hellip;	&#8230;	...	многоточие ...
&prime;	&#8242;	'	одиначный штрих — минуты
&Prime;	&#8243;	"	двойной штрих — секунды

&oline;	&#8254;	–	надчеркивание
&frasl;	&#8260;	/	косая дробная черта
<b>Общая пунктуация</b>			
&ndash;	&#8211;	—	тире
&mdash;	&#8212;	—	длинное тире
&lsquo;	&#8216;	‘	левая одиночная кавычка
&rsquo;	&#8217;	’	правая одиночная кавычка
&sbquo;	&#8218;	,	нижняя одиночная кавычка
&ldquo;	&#8220;	“	левая двойная кавычка
&rdquo;	&#8221;	”	правая двойная кавычка
&bdquo;	&#8222;	„	нижняя двойная кавычка

#### ЛАБОРАТОРНАЯ РАБОТА № 4

1. По представленному образцу на стр. № 42 составить HTML-код.  
По составлению кода предъявляются следующие требования:  
— обязательно указать все необходимые теги и атрибуты для представления указанного образца в HTML-формате;  
— необходимо обязательно указать все служебные символы, которые используются (например, символ знака параграфа — **&#167;** , символ тире — **&#151;** и т.д.  
— формулы оформить в соответствии с требованиями, предъявляемыми к формулам.  
Далее необходимо сформировать внешнюю таблицу стилей со следующими параметрами:
2. Красная строка абзацев — 20px;
3. Выравнивание в основном тексте должно быть по левому краю; у заголовков и подрисуночных надписей — по центру;
4. Отступы от текста слева, справа, сверху и снизу равны 20px.
5. Отступы от рисунка слева, справа — 12 px, сверху и снизу — 10 px. Рисунок оформить в округлой рамке с величиной закругления 10px. Линия рамки — пунктир.
6. Гарнитура шрифта “с засечками”, а размер шрифта основного текста принимается равным 90%, текст заголовков равен 100%, а текст в таблице — 80%.
7. Цвет текста ссылок должен быть — на собственное усмотрение.
8. Цвет текста — на собственное усмотрение.
9. Цвет фона — на собственное усмотрение.
10. Рамка таблицы должна быть толщиной равна 3px.

## § 1. Безопасность и экологичность разработки

**Охрана труда** — это система законодательных актов, социально-экономических, организационных, технических, гигиенических и лечебно-профилактических мероприятий и средств, обеспечивающих безопасность, сохранение здоровья и работоспособности человека в процессе труда.

В проекте разрабатывается «**синтезатор СВЧ**» автоматизированной системы мониторинга источников радиоизлучений.

Разрабатываемое изделие проходит четыре стадии жизни:

- 1) проектирование;
- 2) изготовление и настройка;
- 3) эксплуатация;
- 4) утилизация.

Карта условий труда в лаборатории представлена в [табл. 1](#) (*создать ссылку на таблицу*), где наряду с санитарно-гигиеническими факторами оценим основные психофизиологические факторы, сопутствующие этапу проектирования данного изделия.

**Т а б л и ц а 1**

Наименование фактора	Нормативное значение фактора	Действующее значение фактора
Санитарно-гигиенические факторы		
1 Эффективно-эквивалентная температура воздуха (ЭЭТ) на рабочем месте (в помещении), °С: в теплый период года, в холодный период года	+20...+22 +18...+20	+20.8 +19.1
2 Относительная влажность воздуха, %: в теплый период года, в холодный период года	60...40 60...40	45 60
3 Химические вещества: свинец и его неорганические соединения, мг/м <sup>3</sup>	До 0.01	До 0.01
4 Промышленный шум, дБА	50	43

На рис. 1 (*создать ссылку на рисунок*) приведена схема выносного заземляющего устройства.

Описание схемы лаборатории:

- монтажный стол;
- лабораторные столы;
- вытяжное устройство;
- стулья;
- рабочие столы.

Схема выносного заземляющего устройства

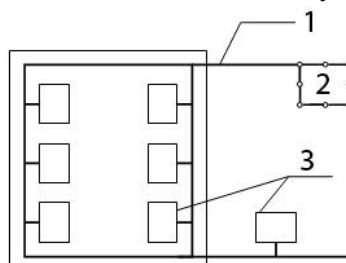


Рис. 1. 1 — заземлитель; 2 — заземляющие проводники; 3 — заземляющее оборудование.

Выберем коэффициент использования светового потока  $V$  по следующим данным:

- коэффициент отражения побеленного потолка,  $R_{\text{п}} = 70\%$ ;
- коэффициент отражения от стен, окрашенных, в светлую краску,  $R_{\text{ст}} = 50\%$ ;
- коэффициент отражения от пола, покрытого темным паркетом или линолеумом  $R_{\text{р}} = 10\%$ ;